INFINITE HORIZON NONLINEAR QUADRATIC COST REGULATOR

A Thesis
Presented to
The Academic Faculty

By

Hassan A. Almubarak

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
School of Electrical and Computer Engineering

Georgia Institute of Technology

December 2018

INFINITE HORIZON NONLINEAR QUADRATIC COST REGULATOR

Approved by:

Dr. Nader Sadegh, Advisor
The George W. Woodruff School of Mechanical Engineering
Georgia Institute of Technology

Dr. David G. Taylor, Co-advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Patricio Antonio Vela
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Samuel Coogan
School of Electrical and Computer Engineering
Georgia Institute of Technology

Date Approved: December 06, 2018

To my first teachers, my parents Fatima and Abdullah

ACKNOWLEDGEMENTS

I would like to highly express my gratitude to my advisor Dr Nader Sadegh for his guidance and enthusiasm throughout my research and my MS journey. Dr Sadegh's office was opened whenever I faced an obstacle or felt a dead end in my research to help resolving the problem. I would like to thank my coadvisor Dr David G. Taylor for his very valuable comments and engagement. I am thankful to my committee members Dr Patricio Antonio Vela, Dr. Samuel Coogan and to all my professors at Georgia Tech and at King Fahd University of Petroleum and Minerals (KFUPM) who have contributed in sharpening my research skills and critical thinking. Furthermore, I would like to thank my family whom I would not be here without their support. I'm grateful to my beloved brothers and sisters and to the key of my strength, happiness and kindness, my parents, my wife Mayamin and my little boy Mujtaba. My loved ones, family and friends, have encouraged me throughout my journey to do better and better. I will be thankful and pleased forever for your love.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Infinite horizon optimal control has been a leading methodology for both linear and nonlinear systems. The Hamilton-Jacobi-Bellman (HJB) approach is a very effective approach for infinite horizon optimal control which involves solving the associated nonlinear partial differential equation known as the HJB equation. Because of the importance and high difficulty of solving the HJB equation, different techniques and approximations to solve the HJB equation are proposed in the literature. In the case of linear systems, the HJB equation becomes the known Reccati equation which provides the well known and powerful Linear Quadratic Regulator (LQR).

Therefore, the focus of this research is to generalize the idea of the LQR and develop a Nonlinear Quadratic cost Regulator (NLQR) based on the solution of the HJB equation for the infinite horizon problem. We present a novel and an efficient technique based on Taylor series expansion for the HJB equation around an equilibrium point. Utilizing a set of minimal polynomial basis functions that includes all possible combinations of the states, a nonlinear matrix equation similar to the Riccati equation is constructed from the HJB equation. Solving this nonlinear matrix equation term by term renders the associated value function (i.e, optimal cost-to-go) and the optimal controller with a prescribed truncation order. The computational complexity of this approach is shown to have only a polynomial growth rate with respect to the series order.

The developed HJB based equation can be solved independently of the current states and hence the optimal nonlinear control can be obtained a-priori offline for smooth nonlinear control affine systems. A general recursive closed form procedure to find the coefficients of high order control laws is provided. Set of examples are presented with different systems natures and nonlinearities including inputs saturation.

CHAPTER 1
INTRODUCTION

1.1 Background and Motivation

In our real-world, control systems are nonlinear. There are systems, however, that can be approximately described by linear differential equations which then are called linear systems. With the advancements in human's knowledge and life, more complex systems are faced and developed. The complexity of such dynamical systems and the advantages of nonlinear controllers result in the need of nonlinear systems analysis and control. For nonlinear systems, based on the problem and the design goals, diverse nonlinear control techniques have been developed such as feedback linearization, gain scheduling, Lyapunov approach techniques, sliding mode control, optimal control techniques, and many others [1]. Conflicting performance objectives and constraints often call for optimizing the trade-off between the conflicting goals.

Optimal control has been a leading methodology for both linear and nonlinear systems to address the optimization requirements quantitively and qualitatively. Optimization problems and consequently optimal controllers can either be formulated with finite or infinite horizon. Short horizon optimal control problems are often solved numerically using standard methods such as the shooting method or even constrained optimization. A very popular method utilizing the short horizon optimal control is Model Predictive Control (MPC) [2, 3, 4]. The MPC solves the optimization problem at each time sample for the short control horizon ahead and continuously updates the solution. Although it is widely used in applications, it is computationally expensive as it requires re-solving the optimization problem at each sampling instant. An infinite horizon optimal controller, on the other hand, only needs to be updated as a function the system's state if a solution can be found

a-priori. A very effective approach for infinite horizon optimal control, is the Hamilton-Jacobi-Bellman (HJB) approach. This method requires solving the associated HJB equation.

1.2 Hamilton-Jacobi-Bellman Equation

A necessary condition for the existence of the optimal solution is the Hamilton-Jacobi Equation (HJE) which is a nonlinear partial differential equation (PDE) resulting from calculus of variations problems. With an optimality principle comparable to Pontryagin's Minimum/Maximum Principle (PMP), an alternative approach to the variational method, named dynamic programming, was explored by Richard Bellman and coworkers in the later 1950s. As a result, the Hamilton-Jacobi-Bellman (HJB) equation was developed. The HJE equation can be derived from the HJB equation. The HJB equation is capable of producing an optimal feedback control for general nonlinear systems as well as providing Lyapunov candidates. Therefore, the HJB equation plays a pivotal role in searching for the optimal control. The HJB equation is a PDE, rather than an ordinary differential equation (ODE). Thus, solving the HJB equation is not a painless task especially for nonlinear systems. Designing an infinite horizon optimal nonlinear control through the HJB optimization problem is a classical, very challenging optimal control problem. For linear systems with quadratic cost functionals, the HJB equation produces the well-known Riccati equation which is then solved to obtain a symmetric positive definite solution that is used in the linear optimal feedback law. The result is a powerful linear controller, known as the Linear Quadratic cost Regulator (LQR). On the other hand, for nonlinear systems, the HJB equation is a nonlinear PDE which is nearly unworkable analytically [5]. For such a crucial problem in optimal control, tremendous work has been done on studying the geometry and approximating the optimal solution of the HJB equation for nonlinear systems.

1.3 Literature Review

In recent decades, extensive investigations have been done in the area of nonlinear optimal control. Thus, various methods have been proposed to obtain an approximate solution to the HJB equation and/or approximation of the optimal control to obtain a feedback control for general nonlinear systems [6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22].

A recent method developed by Sakamoto and van der Schaft [17], uses the stable manifold theory to find an approximated stabilizing solution of the Lagrangian submanifold of the Hamiltonian system which then the nonlinear control is derived from. When solved iteratively, the algorithm is capable of providing an analytical approximation to the gradient of the solution of the HJB equation, which is the Lagrangian of the Hamiltonian system, as well as a numerical approximation. Thus, since this method approximately finds the stabilizing solution of the Lagrangian, it does not find the solution of the HJB directly. One needs to integrate the approximated Lagrangian to get an approximate solution to the HJB equation, i.e. the value function. Experiments and simulations based on the development of this approach such as stabilization of the acrobat system [23], a magnetic levitation system with constraints [19] and an inverted pendulum with input saturation [18] and others [24] show the capability of this algorithm. However, this method contains a lot of estimations and eliminations such as in estimating the radius of convergence of the sequence of a system developed from the Hamiltonian system, which is not an easy task to do in practice and one needs to find the appropriate iterations number. The radius of convergence must be sufficiently small otherwise the sequence will not converge to the stable manifold solution but rather diverge. Moreover, this method suffers from the need for high computational power in order to get accurate solutions, which makes it not desirable for many systems. In addition, it suffers from curse of dimensionality. As a consequence, analytical approximation is not always attainable.

Another approach is to successively approximate the solution of the HJB equation

through iteratively solving a sequence of the Generalized HJB (GHJB) equations. GHJB equations are linear PDEs that approach an approximate solution to the HJB equation starting by a randomly selected feedback admissible control as in [14], [15], [16] and [22]. It is proven that the successive approximation of the GHJB will eventually converge to an approximate solution to the HJB equation [14]. The Galerkin successive approximation method is popular in this approach. Although this method can be used for a wide class of systems, different approximations can be computed and the quality of the solution depends on the initialization of the control. Moreover, this can be a very expensive method as it requires computations of many integrals.

Other methods try to solve the problem at each point in time. A famous algorithm generated from the HJB equation is known as the State-Dependent Riccati Equation (SDRE). The idea is to factorize the system's dynamics to make it look similar to the linear case but with a state dependent matrix. This factorization is also called apparent linearization. By doing so, a state-dependent Riccati equation is developed. A lot of work has been done on the SDRE method as a generalization for the LQ method for nonlinear systems. Several reported real-life applied projects and simulations proved the success of the SDRE method as reported in a great survey on SDRE [25]. On the contrary, the SDRE has drawbacks in its efficiency, stability and uniqueness of the generated solution.

The idea of using power series expansion has been investigated mostly in flight control systems papers. In a very early work, Al'brekht [6] developed and proved a sufficient condition to a nonlinear optimal control for analytic systems and constructed a general systematic procedure to obtain the controller as a power series of the states for a scalar controller. Lukes [7] developed that for finite number of inputs and relaxed the analyticity assumption to twice differentiable. In addition, proof of existence and uniqueness, in a local sense, of the optimal control was provided. Nevertheless, no recursive closed form procedure was provided to obtain the control law. Garrard [9] adopted similar idea and expanded the value function as a power series of an artificial variable around the origin.

The proposed method further developed in [10] and [12] by expanding the system's dynamics as a power series too which is applicable to a wider class of systems. It was tested and compared to the LQR method in automatic flight control systems and proved its superiority. Nishkawa et al. [8] developed a more efficient procedure to find the coefficients by assuming that the artificial variable is sufficiently small to find a sub-optimal control in a power series form of the artificial variable. Wernli and Cook in [11] used the idea of apparent linearization to develop a sub-optimal control by finding a Taylor series to a state-dependent Riccati equation. This is also considered an early attempt and a contribution in development of the SDRE method [25]. In [20], Xin and Balakrishnan proposed a method similar to [11] but improved in an attempt to achieve larger area of convergence of the power series of the artificial variable in the value function. In their approach, the functional is modified by adding a matrix that depends on the nonlinearity of the system to penalize the states. In that event, the matrix adds penalization to higher orders which is designed to die out with time. However, because apparent linearization is not unique and different linearizations result in different approximations, bad or nonconvergent solution is not unexpected. It is suggested to have an apparent linearization that results in a full-rank controllability matrix [25]. As a result, quality of sub-optimality depends on the choice of the apparent linearization. Moreover, finding the optimal factor is very difficult and some conditions need to be satisfied [20]. Following this further, due to its complexity, it may not be applicable for systems with high dimensions or high nonlinearities in addition to the required online computations. Nevertheless, most of these methods do not apply to systems with different nonlinearities and some consider one or two orders of nonlinearity only and ignore higher orders as well explained in [26].

1.4 Outlines

In this thesis, we develop a novel technique to produce a Taylor series expansion for the HJB equation around an equilibrium point and solve it to provide the optimal controller

as a nonlinear function of the state vector. Moreover, a recursive closed form procedure is provided to obtain the control law as well as the value function. The convergence domain of this controller and the value function coincides with that of the Taylor series of the exact solution under the required conditions. As a consequence, many limitations of the previous methods are overcome. This thesis is organized as follows. Chapter 2 contains optimal control background and the problem statement as well as the basic theorems for the development of the nonlinear optimal controller through the HJB equation. In chapter 3, we utilize tensor algebra tools to efficiently represent the value function and the system nonlinearities as a multivariate Taylor series of the state variables leading to a nonlinear matrix equivalent of the HJB equation. We subsequently present an algorithm to solve this matrix equation recursively. The resulting solution is shown to generate the optimal controller as a nonlinear function of the state vector up to a prescribed truncation order under the required conditions. In chapter 4, we present five examples with different systems natures and nonlinearities including constrained actuators. Finally, in chapter 5, we give some insights and conclusions on the results of the proposed algorithm as well as suggesting possible future developments.

CHAPTER 2

INFINITE HORIZON OPTIMAL CONTROL FOR NONLINEAR SYSTEMS

2.1 Optimal Control Background

For the general time-invariant system

$$\dot{x}(t) = f(x(t), u(t))$$

where $f : R^n \to R^n$, it is desired to find a control, $u(t)$, that optimizes the performance of the system quantitatively. For the sake of convenience, we drop the time argument, t, in our development. Mathematically, it is required to find the control that minimizes the cost functional

$$V(x_0, u) = \int_0^\infty (Q(x) + R(u))dt$$

where $x_0 = x(0)$, $Q(x) \succ 0$ and $R(u) \succ 0$. A necessary condition to minimize the functional is that the minimizer u must minimize the Hamiltonian,

$$H(x, V_x^*, u) = Q(x) + R(u) + V_x^*(x)^T f(x, u) \tag{2.1}$$

where $V^*(x) = \min_{u \in L_2(0,\infty)} V(x_0, u)$ and the vector $V_x^* = (\frac{\partial V^*}{\partial x})^T$. Thus, a necessary condition to the optimal control problem is the Hamilton-Jacobi-Bellman (HJB) equation resulted from the dynamic programming approach,

$$HJB := 0 = \min_u \{H(x, V_x^*, u)\} \tag{2.2}$$

Remark. For general optimal control problems, the value function $V^*(x)$ may not be smooth nor continuous. Nevertheless, under certain conditions, smooth or continuous value

function can be found.

In the next section, based on certain assumptions, existence, uniqueness and continuity of the optimal control and the value function are provided.

2.2 Problem Statement and Formulation

Consider the nonlinear control affine system, a form which many systems can be put in,

$$\dot{x} = f(x) + g(x)u \tag{2.3}$$

where $x \in R^n$, $u \in R^m$, $f : R^n \to R^n$ and $g : R^n \to R^{n\times m}$. Equation 2.3 can be written as

$$\dot{x} = f(x) + \sum_{i=1}^{m} g_i(x)u_i \tag{2.4}$$

where u_i is a scalar and $g_i : R^n \to R^n$. We assume that there exists a neighborhood Ω of the origin such that

$\mathbf{A}_1$ $f(x)$ is at least $C^2(\Omega)$ (i.e., twice continuously differentiable on Ω), $f(0) = 0$, and (F_1, G) is stabilizable where $F_1 = \frac{\partial f}{\partial x}(0)$ and $G = g(0)$.

$\mathbf{A}_2$ The system is stabilizable [7] on Ω: There exists a continuous controller $u(x)$, $\forall x \in \Omega$, for which the origin of the system $\dot{x} = f(x) + g(x)u(x)$ is asymptotically stable. We shall refer to such a $u(x)$ as a stabilizing controller.

Our goal is to design a feedback nonlinear controller that minimizes the quadratic cost functional

$$V(x_0, u) = \frac{1}{2}\int_0^\infty (x^T Q x + u^T R u)dt \tag{2.5}$$

where $x_0 = x(0)$, $Q \succ 0$ and $R \succ 0$. It is worth mentioning that the control affine requirement can be relaxed by adding new states as we show how through an example later.

2.2.1 Existence and Uniqueness of the Optimal Solution and Sufficiency of the HJB Equation

The following theorem [7] guarantees the existence and uniqueness of an optimal control solution.

Theorem 1. *For the stabilizable system (2.3) satisfying A_1 and A_2 with cost functional (2.5), there exists a unique $C^1(\Omega)$, continuously differentiable, optimal feedback control $u^*(x)$ that minimizes the Hamiltonian given by*

$$u^*(x) = -R^{-1}g(x)^T V_x^*(x) \tag{2.6}$$

$\forall x \in \Omega$ where $V^*(x) = \min_{u \in L_2(0,\infty)} V(x_0, u) \in C^2(\Omega)$ and $V_x^* = (\frac{\partial V^*}{\partial x})^T$. Moreover, if $f \in C^\omega(\Omega)$ (i.e., real analytic) so are u^* and V^*.

One may refer to [7] for the detailed proof. Since the existence of the optimal controller is established, we need a sufficient condition to the optimal control problem. In order to get to the sufficient condition, first we need the following theorem [5, 27].

Theorem 2. *For the optimal control problem*

$$\dot{x} = f(x) + g(x)u$$
$$V(x_0, u) = \frac{1}{2}\int_0^\infty (x^T Q x + u^T R u) dt$$

under the proposed assumptions, Theorem 1, and the optimal control $u^*(x)$ in (2.6), the origin of the closed loop system $f(x) + g(x)u^*(x)$ is asymptotically stable.

Proof. The HJB equation (2.2) with the optimal control $u^*(x)$ can be rewritten as

$$\frac{1}{2}(x^T Q x + u^{*T}(x) R u^*(x)) + V_x^*(x)^T (f(x) + g(x)u^*(x)) = 0$$

Then, let the value function $V^*(x) > 0 \; \forall x \neq 0 \in C^1(\Omega)$ satisfies the HJB equation. Then,

$$-\frac{1}{2}(x^T Q x + u^{*T}(x) R u^*(x)) = V_x^*(x)^T (f(x) + g(x)u^*(x)) = \frac{dV^*}{dt}$$
$$\Rightarrow \frac{dV^*}{dt} < 0 \; \forall x \neq 0$$

Hence, Lyapunov conditions are satisfied and thus the HJB equation provides a Lyapunov function to the closed loop system $f(x)+g(x)u^*(x)$ so the origin of the closed loop system is asymptotically stable. ■

The reader may refer to [5, 13, 27] for more analysis upon Lyapunov and the HJB. Now, we are in a position to establish following theorem [5, 27] which provides a necessary and sufficient condition to the optimal control problem.

Theorem 3. *Suppose there exists a stabilizing control $u^*(x)$ that minimizes the Hamiltonian, and let the value function $V^*(x) \in C^1(\Omega)$ satisfies the well-known HJB equation for the infinite horizon problem*

$$V_x^{*T} f(x) - \frac{1}{2} V_x^{*T} g(x) R^{-1} g(x)^T V_x^* + \frac{1}{2} x^T Q x = 0 \tag{2.7}$$

with boundary condition $V^*(0) = 0$, then u^* and $V^*(x)$ are the optimal controller and the optimal cost-to-go to the optimal control problem respectively.

Proof. Consider the control u^* in (2.6) that minimizes the Hamiltonian and let the associated optimal trajectory to be x^*. Then, the HJB equation can be written as

$$V_x^{*T}(f(x^*) + g(x^*)u^*) + \frac{1}{2}(x^{*T} Q x^* + u^{*T} R u^*) = 0$$
$$\Rightarrow \frac{dV^*}{dt} = -\frac{1}{2}(x^{*T} Q x^* + u^{*T} R u^*)$$

Now, integrating both sides for the time interval $[0, \infty]$ yields

$$\int_0^\infty \frac{\partial V^*}{\partial t} = -\int_0^\infty \frac{1}{2}(x^{*T}Qx^* + u^{*T}Ru^*)dt$$
$$\Rightarrow V^*(x(\infty)) - V^*(x(0)) = -\int_0^\infty \frac{1}{2}(x^{*T}Qx^* + u^{*T}Ru^*)dt$$

By Theorem 2, the closed loop system $\dot{x} = f(x) + g(x)u^*(x)$ is asymptotically stable so that $x(\infty) := \lim_{t\to\infty} x(t) = 0$. Thus, $V^*(x(\infty)) = 0$ by continuity of $V^*(x)$ and we get

$$V^*(x(0)) = \frac{1}{2}\int_0^\infty (x^{*T}Qx^* + u^{*T}Ru^*)dt$$

Let there exist an arbitrary control u that produces the trajectory $x(t)$. Since u^* in the HJB equation minimizes $V_x^{*T}g(x)u + \frac{1}{2}u^TRu$ with respect to u, we have

$$V_x^{*T}(f(x) + g(x)u) + \frac{1}{2}(x^TQx + u^TRu) \geq 0$$
$$\Rightarrow \frac{\partial V^*}{\partial t} \geq -\frac{1}{2}(x^TQx + u^TRu)$$

Integrating both sides for the interval $[0, \infty]$ and using $V^*(x(\infty)) = 0$ shown earlier, we get

$$V^*(x(\infty)) - V^*(x(0)) \geq -\frac{1}{2}\int_0^\infty \frac{1}{2}(x^TQx + u^TRu)dt$$
$$\Rightarrow V^*(x(0)) \leq V(x(0), u)$$

Hence, the control u^* produces the optimal cost $V^*(x(0))$ and no other controller provides a lower cost. ■

Therefore, the HJB equation is a necessary and a sufficient condition to the optimal control problem (2.3), (2.5). Moreover, in view of Theorem 1 and Theorem 3, a unique solution u^* that minimizes (2.5) subject to (2.3) is attainable. It is also important to note

that the value function V^* is real analytic (i.e., has a convergent Taylor series) if (2.3) is real analytic.

In the next chapter, we will formulate an efficient method for computing the Taylor series of V^* to within a prescribed order. Additionally, a closed form solution to find the coefficients of an arbitrary order will be provided.

CHAPTER 3

INFINITE HORIZON NONLINEAR QUADRATIC COST REGULATOR

3.1 Development of the Nonlinear Quadratic Cost Regulator

The idea is to compute the value function with the first $\bar{k}$ terms of its Taylor series using tensor algebra as

$$V^*(x) = \frac{1}{2}x^T\bar{P}_1x + \frac{1}{3}x^T\bar{P}_2x^{\otimes 2} + \frac{1}{4}x^T\bar{P}_3x^{\otimes 3} + \ldots + \frac{1}{\bar{k}+1}x^T\bar{P}_{\bar{k}}x^{\otimes \bar{k}} + O(x^{\bar{k}+2})$$

$$\Rightarrow V^*(x) = \sum_{k=1}^{\bar{k}} \frac{x^T\bar{P}_kx^{\otimes k}}{k+1} + O(x^{\bar{k}+2}) \tag{3.1}$$

where $\bar{P}_k \in \mathbb{R}^{n\times n^k}$ is a matrix representation of a symmetric tensor of rank k, $\otimes$ is the Kronecker product, and $x^{\otimes k} = x \otimes x \ldots \otimes x$ k times. We can view $\bar{P}_k$ as matricization of a tensor by unfolding it. The following Kronecker product properties and equations will be used throughout the development [28]. For the matrices A, B, C and the scalars a, $a_1, \ldots a_n$

1. A $\otimes(B + C) = A \otimes B + A \otimes C$.
2. $a \otimes A = aA = Aa = A \otimes a$.
3. $(A \otimes B)^T = A^T \otimes B^T$.
4. $(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}$, for non-singular A and B.
5. vec$(ABC) = (C^T \otimes A)$vec(B), where vec is an operator such that vectorizing A

yields $\text{vec}(A_{m\times n}) = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix}$.

The result of the Kronecker product of the states vector creates repeated basis which results in dependent vectors in the tensor $\bar{P}_k$. In spite of its high dimension, $x^{\otimes k} \in n^k$ has many redundant terms that can be reduced to a minimal vector x^k of dimension $m_k = \binom{n+k-1}{k}$, which has a polynomial growth rate ($O(k^n)$) for a fixed n. As an example, consider a second order system ($n = 2$), with $x^{\otimes 2}$ given by $x \otimes x = \begin{bmatrix} x_1^2 \\ x_1x_2 \\ x_2x_1 \\ x_2^2 \end{bmatrix}$. The corresponding tensor $\bar{P}_2$ can be viewed as $\bar{P}_2 = \left[\begin{array}{cc|cc} a & \frac{b}{2} & \frac{b}{2} & e \\ c & \frac{d}{2} & \frac{d}{2} & f \end{array}\right]$. The unique L_2 to produce $x^2 = \begin{bmatrix} x_1^2 \\ x_2x_1 \\ x_2^2 \end{bmatrix}$ is $L_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$: $x^{\otimes 2} = L_2x^2$ and $P_2 = \bar{P}_2L_2 = \begin{bmatrix} a & b & e \\ c & d & f \end{bmatrix}$. Generally, $x^{\otimes k} = L_kx^k$ and $P_k = \bar{P}_kL_k$ where $L_k \in \mathbb{R}^{n\times m_k}$ is a unique constant matrix mapping the reduced vector x^k to $x^{\otimes k}$. Substituting this into (3.1) leads to the value function

$$V^*(x) = \sum_{k=1}^{\bar{k}} \frac{x^T(\bar{P}_kL_k)x^k}{k+1} + O(x^{\bar{k}+2})$$

$$\Rightarrow V^*(x) = \sum_{k=1}^{\bar{k}} \frac{x^TP_kx^k}{k+1} + O(x^{\bar{k}+2}) \tag{3.2}$$

Then, the Jacobian of the k–term, $V_k(x) = \frac{1}{k+1}x^T P_k x^k$, generated by a symmetric tensor can be seen to be

$$\left(\frac{\partial V_k}{\partial x}\right)^T = P_k x^k \tag{3.3}$$

The following proposition provides the necessary and sufficient condition on P_k in order for (3.3) to be valid in general.

Proposition 4. *The relationship given by (3.3) holds for an arbitrary P_k if and only if the Jacobian of $P_k x^k$ is symmetric.*

Proof. If (3.3) holds then the Jacobian of $P_k x^k$, being the Hessian of V_k, must be symmetric. To prove the converse suppose that the Jacobian of $P_k x^k$ is symmetric. Then

$$\begin{aligned}(k+1)\left(\frac{\partial V_k}{\partial x}\right)^T &= P_k x^k + \left(P_k \frac{\partial x^k}{\partial x}\right)^T x \\ &= P_k x^k + P_k \frac{\partial x^k}{\partial x} x = (k+1) P_k x^k\end{aligned}$$

where the last equation follows from

$$\frac{\partial x^k}{\partial x} x = \sum_{i=1}^{n} \frac{\partial x^k}{\partial x_i} x_i = k x^k$$

thus completing the proof. ■

Assuming that P_k's satisfy the symmetry requirement, the gradient of the value function in (3.2) may be expressed as

$$V_x^*(x) = P_1 x + P_2 x^2 + \ldots + P_{\bar{k}} x^{\bar{k}} + O(x^{\bar{k}+1}) = \sum_{k=1}^{\bar{k}} P_k x^k + O(x^{\bar{k}+1})$$

which can be written compactly as

$$V_x^*(x) = \boldsymbol{P}\xi(x) + O(x^{\bar{k}+1}) \tag{3.4}$$

where the compact notations $\boldsymbol{P} = [P_1 \ \dots \ P_{\bar{k}}] \in R^{n\times N}$ and $\xi = \begin{bmatrix} x \\ x^2 \\ \vdots \\ x^{\bar{k}} \end{bmatrix} \in R^N$, $N = \sum_{k=1}^{\bar{k}} m_k = \binom{n+\bar{k}}{n} - 1$, will be used throughout. The resulting optimal controller may be obtained up to order $\bar{k}$ by substituting (3.4) into (2.6):

$$u^*(x) = -R^{-1}g(x)^T\boldsymbol{P}\xi(x) + O(x^{\bar{k}+1}) \tag{3.5}$$

Meanwhile, the system's vector field using the same basis functions may be expressed up to order $\bar{k}$ by

$$f(x) = [F_1 \ \dots \ F_{\bar{k}}][x^T \ \dots \ x^{\bar{k}T}]^T = F\xi(x) + O(x^{\bar{k}+1}) \tag{3.6}$$

for some $F \in R^{n\times N}$. Substituting for u^* and f from the preceding equations into the HJB equation (2.7) will result in

$$\xi^T(x)\boldsymbol{P}^T F\xi(x) - \frac{1}{2}\xi^T(x)\boldsymbol{P}^T g(x)R^{-1}g(x)^T\boldsymbol{P}\xi(x) + \frac{1}{2}x^T Qx = O(x^{\bar{k}+2})$$

Since the linear state x is a part of the basis vector $\xi(x)$, x^TQx can be replaced with $\xi(x)^T\bar{Q}\xi(x)$ where $\bar{Q} \in R^{N\times N}$ and is all zeros matrix except for the first n by n submatrix being Q. Then,

$$\xi^T(x)\boldsymbol{P}^T F\xi(x) - \frac{1}{2}\xi^T(x)\boldsymbol{P}^T g(x)R^{-1}g(x)^T\boldsymbol{P}\xi(x) + \frac{1}{2}\xi(x)^T\bar{Q}\xi(x) = O(x^{\bar{k}+1})$$

$$\Rightarrow \xi(x)^T(2\boldsymbol{P}^T F - \boldsymbol{P}^T g(x)R^{-1}g(x)^T\boldsymbol{P} + \bar{Q})\xi(x) = O(x^{\bar{k}+1}) \tag{3.7}$$

One can see that nonquadratic cost may also be applicable through this method. In addition,

it is worth noting that if the matrix $g(x)$ in (2.3) is a constant matrix then the HJB equation (3.7) will be

$$\xi(x)^T(2\boldsymbol{P}^T F - \boldsymbol{P}^T G R^{-1} G^T \boldsymbol{P} + \bar{Q})\xi(x) = O(x^{\bar{k}+1}) \tag{3.8}$$

Notice that (3.8) looks very similar to the Riccati equation except that the unknown matrix $\boldsymbol{P}$ happens to be non-square. In fact, if our series truncation order is one (i.e., linear basis), this would lead to the well-known Algebraic Riccati Equation (ARE). We will explore the solution of (3.7) in the next section but before that we want to propose the following assumption which makes the computations easier. The positive-definite matrix R in (2.5) is diagonal. This is a fair assumption since one can easily diagonlize the positive-definite matrix and define a new control variable through a linear mapping. As an illustration, consider the general positive-definite matrix $\bar{R}$ that penalizes an input v. Then, the quadratic cost functional is given by

$$V(x_0, u) = \frac{1}{2}\int_0^\infty (x^T Q x + v^T \bar{R} v) dt$$

The positive definite matrix $\bar{R}$ can be decomposed to VRV^T. Moreover, one can definitely decompose $\bar{R}$ such that R is an identity matrix, i.e. $\bar{R} = VV^T$. However, we will consider the general case as in many cases control designers choose R to be a diagonal matrix. Then the functional will be

$$V(x_0, u) = \frac{1}{2}\int_0^\infty (x^T Q x + v^T V R V^T v) dt$$

Now define the vector $u = V^T v$. Hence, the functional becomes

$$V(x_0, u) = \frac{1}{2}\int_0^\infty (x^T Q x + u^T R u) dt$$

where R is a diagonal matrix and of course in a such case one needs to make the necessary changes in the system's dynamics. Therefore, the i^{th} optimal control input in the system

can be found by

$$u_i{}^*(x) = -r_i^{-1} g_i(x)^T \boldsymbol{P} \xi(x) + O(x^{\bar{k}+1})$$

Consequently, the HJB equation in (3.7) can be described as

$$\xi(x)^T (2\boldsymbol{P}^T F - \boldsymbol{P}^T \sum_{i=1}^{m} (g_i(x) r_i^{-1} g_i(x)^T) \boldsymbol{P} + \bar{Q}) \xi(x) = O(x^{\bar{k}+2}) \tag{3.9}$$

Moreover, each g_i can be expanded up to order $\bar{k}$ by

$$g_i(x) = G_{i0} + [G_{i1} \ \dots \ G_{i\bar{k}}][x^T \ \dots \ x^{\bar{k}T}]^T = G_{i0} + G_i \xi(x) + O(x^{\bar{k}+1}) \tag{3.10}$$

for some $G_i \in R^{n \times N}$ and a constant vector $G_{i0} \in R^n$. Accordingly (3.9) becomes,

$$\xi(x)^T (2\boldsymbol{P}^T F - \boldsymbol{P}^T \sum_{i=1}^{m} ((G_{i0} + G_i \xi(x)) r_i^{-1} (\xi(x)^T G_i^T + G_{i0}^T)) \boldsymbol{P} + \bar{Q}) \xi(x) = O(x^{\bar{k}+2}) \tag{3.11}$$

$$\begin{aligned} \Rightarrow \xi(x)^T (2\boldsymbol{P}^T F - \boldsymbol{P}^T \sum_{i=1}^{m} (G_{i0} r_i^{-1} G_{i0}^T) \boldsymbol{P} - \boldsymbol{P}^T \sum_{i=1}^{m} (G_{i0} r_i^{-1} \xi(x)^T G_i^T) \boldsymbol{P} \\ - \boldsymbol{P}^T \sum_{i=1}^{m} (G_i \xi(x) r_i^{-1} G_{i0}^T) \boldsymbol{P} - \boldsymbol{P}^T \sum_{i=1}^{m} (G_i \xi(x)) r_i^{-1} (\xi(x)^T G_i^T) \boldsymbol{P} + \bar{Q}) \xi(x) = O(x^{\bar{k}+2}) \end{aligned}$$

Note that the term $\boldsymbol{P}^T \sum_{i=1}^{m} (G_{i0} r_i^{-1} G_{i0}^T) \boldsymbol{P}$ can be compacted to be $\boldsymbol{P}^T G R^{-1} G \boldsymbol{P}$. Hence, the HJB equation can be further expressed as

$$\begin{aligned} \xi(x)^T (2\boldsymbol{P}^T F - \boldsymbol{P}^T G R^{-1} G - \sum_{i=1}^{m} (G_{i0} r_i^{-1} \xi(x)^T G_i^T) - \sum_{i=1}^{m} (G_i \xi(x) r_i^{-1} G_{i0}^T) \\ - \sum_{i=1}^{m} (G_i \xi(x)) r_i^{-1} (\xi(x)^T G_i^T) \boldsymbol{P} + \bar{Q}) \xi(x) = O(x^{\bar{k}+2}) \end{aligned} \tag{3.12}$$

or

$$\xi(x)^T(2\boldsymbol{P}^T F - \boldsymbol{P}^T \sum_{k=0}^{\bar{k}} \sum_{i=1}^{m} (g_i(x^k) r_i^{-1} g_i^T(x^k))\boldsymbol{P} + \bar{Q})\xi(x) = O(x^{\bar{k}+2}) \tag{3.13}$$

Now, we are in a good shape to develop an efficient algorithm and a closed form solution of a tensor P_k so then we can evaluate the value function up to a desired order of approximation.

3.2 Solving the HJB Equation

The HJB equation (3.7) is a nonlinear matrix equation in terms of the unknown matrix $\boldsymbol{P}$. It might be tempting to solve the equation by simply setting the inner matrix to zero while ignoring ξ. Such a simplistic approach will fail as $\xi^T \boldsymbol{M} \xi = 0$ does not in general imply that $\boldsymbol{M} = 0$. Instead, the proposed algorithm solves this equation sequentially by setting the coefficients of various powers of x to zero. Before discussing the algorithm details, we need to find a way to enforce the symmetry condition given by Proposition 4 independently of x.

Proposition 5. *Matrix $P_k \in R^{n \times m_k}$ satisfies the symmetry condition given by Proposition 4 if and only if $S_k \mathrm{vec}(P_k) = 0$ for a constant full rank $S_k \in \mathbb{R}^{nm_k - m_{k+1} \times m_k}$.*

Proof. Let e_i, $i = 1, \ldots, n$ be the canonical basis (e.g., $[1\ 0]^T$ and $[0\ 1]^T$ for $n = 2$). Then the symmetry condition on P_k is equivalent to $e_i^T P_k \frac{\partial x^k}{\partial x_j} - e_j^T P_k \frac{\partial x^k}{\partial x_i} = 0$. Each $\frac{\partial x^k}{\partial x_i}$, being of order $k-1$, can be represented as $N_i x^{k-1}$, $i = 1, \ldots n$, for a constant matrix $N_i \in \mathbb{R}^{m_k \times m_k}$. Thus $(e_i^T P_k N_j - e_j^T P_k N_i) x^{k-1} = 0$, $\forall x \in \mathbb{R}^n$, implying that $e_i^T P_k N_j - e_j^T P_k N_i = 0$. Vectorizing both sides, yields

$$(e_i \otimes N_j - e_j \otimes N_i)^T \mathrm{vec}(P_k) = 0$$

Let $Q_k \in R^{n \times m_k}$ be an arbitrary matrix and consider $f_k(x) = x^T Q_k x^k$. The Jacobian

$\partial f_k/\partial x$ of f_k is necessarily of the form $P_k x^k$ for a unique P_k (depending on Q_k) satisfying the symmetry condition. Furthermore, $x^T(Q_k - P_k)x^k = 0$ implying that $K(\mathrm{vec}(Q_k) - \mathrm{vec}(P_k)) = 0$ where K is $nm_k \times m_{k+1}$ matrix of rank m_{k+1} such that $x^k \otimes x = Kx^{k+1}$. From the existence and uniqueness of P_k for an arbitrary Q_k, it follows that the column space

$$\bigcup_{i,j>i} \mathrm{col}(e_i \otimes N_j - e_j \otimes N_i) \cup \mathrm{col}(K^T) = \mathbb{R}^{nm_k}$$

The proof is established by choosing $nm_k - m_{k+1}$ linearly independent $\{w_j\}_{j=1}^{nm_k - m_{k+1}}$ vectors from $\bigcup_{i,j>i} \mathrm{col}(e_i \otimes N_j - e_j \otimes N_i)$ and setting $S_k = [w_1, w_2, \ldots, w_{nm_k - m_{k+1}}]^T$. ■

We are now in a position to fully describe our algorithm for finding the components of the value function V^* and the optimal controller u^*. This is accomplished by setting the matrix coefficients of each term x^k in the Taylor series expansion of the HJB equation (3.9) to zero. The first term of the series renders P_1 by solving a standard Riccati equation. The subsequent P_k's are computed recursively based on P_1, G_{ik}'s, F_k's and the previous P_k's.

3.3 NLQR Algorithm Details

Given f, g, $\bar{Q}$, R, and $\bar{k}$, implement the following steps to find P_k, $k = 1, \ldots, \bar{k}$, used to compute $V^*(x)$ and $u^*(x)$:

1. Compute the matrices components F_k of $f(x) = \sum_{k=1}^{\bar{k}} F_k x^k + O(x^{\bar{k}+1})$ and G_{ik} of $g_i(x) = \sum_{k=1}^{\bar{k}} G_{ik} x^k + O(x^{\bar{k}+1})$ for $k = 1, \ldots, \bar{k}$, and $i = 1, \ldots, m$.

2. For the quadratic order of the value function, i.e. linear control case, in (3.7), which corresponds to $k = 1$, solve the Riccati equation

$$F_1^T P_1 + P_1 F_1 - P_1 G R^{-1} G^T P_1 + \bar{Q}_{n\times n} = 0$$

for P_1. This equation is guaranteed to have a symmetric positive definite solution by Assumption A_1.

3. For higher orders, using (3.12), we take all possible combinations of terms with same power. Thus, for the cubic order of the value function, which corresponds to $k = 2$, we have

$$x^T 2P_1F_2x^2 - x^T P_1GR^{-1}G^TP_2x^2 - x^T P_1 \sum_{i=1}^{m}(G_{i0}r_i^{-1}x^TG_{i1}^T)P_1x + x^T\bar{Q}_{n\times n_2}x^2 +$$
$$x^{2T}2P_2^TF_1x - x^{2T}P_2^TGR^{-1}G^TP_1x - x^TP_1\sum_{i=1}^{m}(G_{i1}xr_i^{-1}G_{i0}^T)P_1x + x^{2T}\bar{Q}_{n_2\times n}x = 0$$

Since $\bar{Q}$ is all zeros matrix except for the first n by n elements, then $\bar{Q}_{n\times n_2} = 0$ and $\bar{Q}_{n_2\times n} = 0$. In addition, considering the fact that each term is a scalar, let us take the transpose of some of the terms to get

$$x^T[2P_1F_2 + 2F_1^TP_2 - 2P_1GR^{-1}G^TP_2]x^2 - 2x^TP_1\sum_{i=1}^{m}(G_{i0}r_i^{-1}x^TG_{i1}^T)P_1x = 0 \quad (3.14)$$

Now, we need to combine the two terms by modifying the second term to have x^2 in its right. As an illustration, we will consider one of the g_i's, i.e. one input case we consider $x^TG_1^TP_1x$ where we will use vectorization with Kronecker product to get $(x^T \otimes x^T)\text{vec}(G_1^TP_1) = x^{\otimes 2T}\text{vec}(G_1^TP_1)$. Note that the Kronecker product will create some repeated basis which we need to remove without affecting the solution. For example, consider a second order system where we have $x^T = \begin{bmatrix} x_1, x_2 \end{bmatrix}$. The resultant Kronecker will be $x^T \otimes x^T = \begin{bmatrix} x_1^2, x_1x_2, x_2x_1, x_2^2 \end{bmatrix}$ and we want $x^{2T} = \begin{bmatrix} x_1^2, x_1x_2, x_2^2 \end{bmatrix}$. A linear mapping would be

$$x^T \otimes x^T = x^{2T}K_{11}$$

where $K_{11} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$. Then, we get

$$x^{\otimes 2T}\text{vec}(G_1^TP_1) = x^{2T}K_{11}\text{vec}(G_1^TP_1) = \text{vec}^T(G_1^TP_1)K_{11}^Tx^2$$

Hence, equation (3.14) will be

$$x^T[P_1F_2 + F_1^TP_2 - P_1GR^{-1}G^TP_2 - P_1\sum_{i=1}^{m}(G_{i0}r_i^{-1}\text{vec}^T(G_1^TP_1)K_{11}^T)]x^2 = 0$$

$$x^T[P_1F_2 + (F_1^T - P_1GR^{-1}G^T)P_2 - P_1\sum_{i=1}^{m}(G_{i0}r_i^{-1}\text{vec}^T(G_1^TP_1)K_{11}^T)]x^2 = 0 \quad (3.15)$$

Interestingly, we have a negative feedback from the first case multiplied by the unknown matrix. So let

$$F_c = F_1^T - P_1GR^{-1}G^T$$

Now, again we use vectorization with Kronecker product in (3.15) to get

$$x^{2T} \otimes x^T\text{vec}([P_1F_2 - P_1\sum_{i=1}^{m}(G_{i0}r_i^{-1}\text{vec}^T(G_1^TP_1)K_{11}^T) + F_cP_2]) = 0$$

Again, the Kronecker product will create some repeated basis. Now, for a second order system where $x^{2T} = \begin{bmatrix} x_1^2, x_1x_2, x_2^2 \end{bmatrix}$, the resultant Kronecker will be $(x^2)^T \otimes x^T = \begin{bmatrix} x_1^3, x_1^2x_2, x_1x_2^2, x_1x_2^2, x_1x_2^2, x_2^3 \end{bmatrix}$ and we want $x^{3T} = \begin{bmatrix} x_1^3, x_1^2x_2, x_1x_2^2, x_2^3 \end{bmatrix}$. A linear transformation can be found through

$$x^{2T} \otimes x^T = x^{3T}K_{21}$$

which produces

$$K_{21} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Thus,

$$x^{3T}K_{21}\text{vec}(P_1F_2 - P_1\sum_{i=1}^{m}(G_{i0}r_i^{-1}\text{vec}^T(G_1^TP_1)K_{11}^T) + F_cP_2) = 0$$

$$\Rightarrow K_{21}\text{vec}(P_1F_2 - P_1\sum_{i=1}^{m}(G_{i0}r_i^{-1}\text{vec}^T(G_1^TP_1)K_{11}^T) + F_cP_2) = 0$$

Now, using Kronecker product identities we get

$$K_{21}(I \otimes F_c)\text{vec}(P_2) + K_{21}\text{vec}(P_1F_2 - P_1\sum_{i=1}^{m}(G_{i0}r_i^{-1}\text{vec}^T(G_1^TP_1)K_{11}^T)) = 0$$

We also utilize the fact that enforcing the symmetry condition given by Proposition 3 leads to matrix S_2 such that $S_2\text{vec}(P_2) = 0$, which we use to generate more equations to solve for the unknown elements in P_2. This is an illustrative example on how to find S_2 which can be generalized to find S_k. For the second order case $x^{2T} = \begin{bmatrix} x_1^2, x_1x_2, x_2^2 \end{bmatrix}$ and $P_2 = \begin{bmatrix} a & b & e \\ c & d & f \end{bmatrix}$. The symmetry condition in this case requires $b = c$ and $e = d$. Then, $\text{vec}(P_2) = \begin{bmatrix} a \\ c \\ b \\ d \\ e \\ f \end{bmatrix} = \begin{bmatrix} a \\ b \\ b \\ d \\ d \\ f \end{bmatrix}$. Therefore, S_2 can be chosen to be

$$S_2 = \begin{bmatrix} 0 & 1 & -0.5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.5 & -1 & 0 \end{bmatrix} \Rightarrow S_2\text{vec}(P_2) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Hence, we get

$$\begin{bmatrix} K_{21}(I \otimes F_c) \\ S_2 \end{bmatrix} \text{vec}(P_2) = \begin{bmatrix} -K_{21}\text{vec}(P_1F_2 - P_1\sum_{i=1}^{m}(G_{i0}r_i^{-1}\text{vec}^T(G_1^TP_1)K_{11}^T)) \\ 0 \end{bmatrix}$$

$$\Rightarrow \text{vec}(P_2) = \begin{bmatrix} K_{21}(I \otimes F_c) \\ S_2 \end{bmatrix}^{-1} \begin{bmatrix} -K_{21}\text{vec}(P_1F_2 - P_1\sum_{i=1}^{m}(G_{i0}r_i^{-1}\text{vec}^T(G_1^TP_1)K_2^T)) \\ 0 \end{bmatrix}$$

Finally, we reshape $\text{vec}(P_2)$ to get P_2.

4. For the quartic order, which corresponds to $k = 3$, we have

$$\begin{aligned} x^T[2P_1F_3 - P_1GR^{-1}G^TP_3]x^3 - x^TP_1\sum_{i=1}^{m}(G_{i0}r_i^{-1}x^TG_{i1}^T)P_2x^2 - x^TP_1\sum_{i=1}^{m}(G_{i0}r_i^{-1}x^{2T}G_{i2}^T)P_1x \\ +x^{3T}[2P_3^TF_1 - P_3^TGR^{-1}G^TP_1]x - x^{2T}P_2^T\sum_{i=1}^{m}(G_{i1}xr_i^{-1}G_{i0}^T)P_1x - x^TP_1\sum_{i=1}^{m}(G_{i2}x^2r_i^{-1}G_{i0}^T)P_1x \\ +x^{2T}[2P_2^TF_2 - P_2^TGR^{-1}G^TP_2]x^{2T} - x^TP_1\sum_{i=1}^{m}(G_{i1}xr_i^{-1}G_{i0}^T)P_2x^2 \\ -x^TP_1\sum_{i=1}^{m}(G_{i1}xr_i^{-1}x^TG_{i1}^T)P_1x - x^{2T}P_2^T\sum_{i=1}^{m}(G_{i0}r_i^{-1}x^TG_{i1}^T)P_1x = 0 \end{aligned}$$

We must mention that we will need to introduce more linear mapping matrices K as more combinations of higher powers are involved to form the desired power. Going through the same procedure in step 3 leads to

$$x^T[2P_1F_3 - P_1GR^{-1}G^TP_3 - P_1\sum_{i=1}^{m}(G_{i0}r_i^{-1}\text{vec}^T(G_{i1}^TP_2)K_{21}^T)$$
$$-P_1\sum_{i=1}^{m}(G_{i0}r_i^{-1}\text{vec}^T(G_{i2}^TP_1)K_{21}^T)]x^3 + x^{3T}[2P_3^TF_1 - P_3^TGR^{-1}G^TP_1$$
$$-\sum_{i=1}^{m}(K_{21}\text{vec}(G_{i1}^TP_2)r_i^{-1}G_{i0}^T)P_1 - \sum_{i=1}^{m}(K_{21}\text{vec}(G_{i2}^TP_1)r_i^{-1}G_{i0}^T)P_1]x$$
$$+x^{2T}[2P_2^TF_2 - P_2^TGR^{-1}G^TP_2 - \sum_{i=1}^{m}(K_{11}\text{vec}(G_{i1}^TP_1)r_i^{-1}G_{i0}^T)P_2$$
$$-\sum_{i=1}^{m}(K_{11}\text{vec}(G_{i1}^TP_1)r_i^{-1}\text{vec}^T(G_{i1}^TP_1)K_{11}^T) - P_2^T\sum_{i=1}^{m}(G_{i0}r_i^{-1}\text{vec}^T(G_{i1}^TP_1)K_{11}^T)]x^2 = 0$$

After simplifications

$$x^T[2P_1F_3 - P_1GR^{-1}G^TP_3 - P_1\sum_{j=1}^{2}\sum_{i=1}^{m}(G_{i0}r_i^{-1}\text{vec}^T(G_{ij}^TP_{k-j})K_{21}^T)]x^3$$
$$+x^{3T}[2P_3^TF_1 - P_3^TGR^{-1}G^TP_1 - \sum_{j=1}^{2}\sum_{i=1}^{m}(K_{21}\text{vec}(G_{ij}^TP_{k-j})r_i^{-1}G_{i0}^T)P_1]x$$
$$+x^{2T}[2P_2^TF_2 - P_2^TGR^{-1}G^TP_2 - \sum_{i=1}^{m}(K_{11}\text{vec}(G_{i1}^TP_1)r_i^{-1}G_{i0}^T)P_2$$
$$-\sum_{i=1}^{m}(K_{11}\text{vec}(G_{i1}^TP_1)r_i^{-1}\text{vec}^T(G_{i1}^TP_1)K_{11}^T) - P_2^T\sum_{i=1}^{m}(G_{i0}r_i^{-1}\text{vec}^T(G_{i1}^TP_1)K_{11}^T)]x^2 = 0$$

$$\Rightarrow x^T[2P_1F_3 + 2F_1^TP_3 - 2P_1GR^{-1}G^TP_3 - 2P_1\sum_{j=1}^{2}\sum_{i=1}^{m}(G_{i0}r_i^{-1}\text{vec}^T(G_{ij}^TP_{k-j})K_{21}^T)]x^3$$
$$+x^{2T}[2P_2^TF_2 - P_2^TGR^{-1}G^TP_2 - \sum_{i=1}^{m}(K_{11}\text{vec}(G_{i1}^TP_1)r_i^{-1}G_{i0}^T)P_2$$
$$-\sum_{i=1}^{m}(K_{11}\text{vec}(G_{i1}^TP_1)r_i^{-1}\text{vec}^T(G_{i1}^TP_1)K_{11}^T) - P_2^T\sum_{i=1}^{m}(G_{i0}r_i^{-1}\text{vec}^T(G_{i1}^TP_1)K_{11}^T)]x^2 = 0$$

$$\Rightarrow x^T[2P_1F_3 + 2F_cP_3 - 2P_1\sum_{j=1}^{2}\sum_{i=1}^{m}(G_{i0}r_i^{-1}\text{vec}^T(G_{ij}^TP_{k-j})K_{21}^T)]x^3$$
$$+x^{2T}[2P_2^TF_2 - P_2^TGR^{-1}G^TP_2 - 2P_1\sum_{i=1}^{m}(K_{11}\text{vec}(G_{i1}^TP_1)r_i^{-1}G_{i0}^T)P_2$$
$$-\sum_{i=1}^{m}(K_{11}\text{vec}(G_{i1}^TP_1)r_i^{-1}\text{vec}^T(G_{i1}^TP_1)K_{11}^T)]x^2 = 0$$

$$\Rightarrow x^{4T}[K_{31}\text{vec}(2P_1F_3 + 2(I\otimes F_c)\text{vec}(P_3) - 2P_1\sum_{j=1}^{2}\sum_{i=1}^{m}(G_{i0}r_i^{-1}\text{vec}^T(G_{ij}^TP_{k-j})K_{21}^T))$$
$$+K_{22}\text{vec}(2P_2^TF_2 - P_2^TGR^{-1}G^TP_2 - 2P_1\sum_{i=1}^{m}(K_{11}\text{vec}(G_{i1}^TP_1)r_i^{-1}G_{i0}^T)P_2$$
$$-\sum_{i=1}^{m}(K_{11}\text{vec}(G_{i1}^TP_1)r_i^{-1}\text{vec}^T(G_{i1}^TP_1)K_{11}^T)) = 0$$

where the linear maps K_{31} and K_{22} are from $x^T\otimes x^{3T} = x^{4T}K_{31}$ and $x^{2T}\otimes x^{2T} = x^{4T}K_{22}$ respectively. Then, with the equation $S_3\text{vec}(P_3) = 0$ we get

$$\text{vec}(P_3) = \begin{bmatrix} 2K_{31}(I\otimes F_c) \\ S_3 \end{bmatrix}^{-1}$$
$$\begin{bmatrix} -2K_{31}\text{vec}(P_1F_3 - P_1\sum_{j=1}^{2}\sum_{i=1}^{m}(G_{i0}r_i^{-1}\text{vec}^T(G_{ij}^TP_{k-j})K_{21}^T)) \\ -K_{22}\text{vec}(2P_2^TF_2 - P_2^TGR^{-1}G^TP_2 - 2P_1\sum_{i=1}^{m}(K_{11}\text{vec}(G_{i1}^TP_1)r_i^{-1}G_{i0}^T)P_2 \\ -\sum_{i=1}^{m}(K_{11}\text{vec}(G_{i1}^TP_1)r_i^{-1}\text{vec}^T(G_{i1}^TP_1)K_{11}^T)) \\ 0 \end{bmatrix}$$

Finally, reshape $\text{vec}(P_3)$ to get P_3.

5. We continue with same procedure in step 3 and 4 up to the desired power. Before giving the general formulation to get P_k let us define the following matrices that result from

summations of similar order terms

$$A_i = \sum_{j=1}^{k-1} \left(\text{vec}^T (G_{ij}^T P_{k-j}) K_{(k-j)j}^T \right)$$

$$B_i = \sum_{j=1}^{p-1} \left(\text{vec}^T (G_{ij}^T P_{p-j}) K_{p-j,j}^T \right)$$

and

$$C_i = \sum_{h=1}^{s-1} \left(K_{(s-h)j} \text{vec}^T (G_{ih}^T P_{s-h}) \right)$$

Thus, the general formulation would be

$$\text{vec}(P_k) = \begin{bmatrix} 2K_{k1}(I \otimes F_c) \\ S_k \end{bmatrix}^{-1} \begin{bmatrix} -2K_{k1}\text{vec}\left(P_1 F_k - P_1 \sum_{i=1}^{m} \left(G_{i0} r_i^{-1} \sum_{j=1}^{k-1} A_i \right) \right) \\ -\sum_{s=2}^{k-1} \left(K_{sp} \text{vec}\left(2P_s^T F_p - P_s^T G R^{-1} G^T P_p - 2P_s^T \sum_{i=1}^{m} \left(G_{i0} r_i^{-1} B_i \right) \right. \right. \\ \left. \left. - \sum_{i=0}^{m} \left(C_i r_i^{-1} B_i \right) \right) \right) \\ 0 \end{bmatrix} \tag{3.16}$$

where $p = k + 1 - s$, and K_{ij} maps x^{i+j} to $x^i \otimes x^j = K_{ij} x^{i+j}$. For the special case where

the system has a scalar input, i.e. a single input system, as in [6], equation (3.16) will be

$$\text{vec}(P_k) = \begin{bmatrix} 2K_{k1}(I \otimes F_c) \\ S_k \end{bmatrix}^{-1} \begin{bmatrix} -2K_{k1}\text{vec}\left(P_1F_k - P_1GR^{-1}\sum_{j=1}^{k-1}\left(\text{vec}^T(G_j^TP_{k-j})K_{(k-j)j}^T\right)\right) \\ -\sum_{s=2}^{k-1}\left(K_{sp}\text{vec}\left(2P_s^TF_p - P_s^TGR^{-1}G^TP_p - 2P_s^TGR^{-1}\sum_{j=1}^{p-1}\left(\text{vec}^T(G_j^TP_{p-j})K_{(p-j)j}^T\right)\right.\right. \\ \left.\left. -\sum_{h=1}^{s-1}\left(K_{(s-h)j}\text{vec}^T(G_h^TP_{s-h})\right)R^{-1}\sum_{j=1}^{p-1}\left(\text{vec}^T(G_j^TP_{p-j})K_{(p-j)j}\right)\right)\right) \\ 0 \end{bmatrix} \tag{3.17}$$

or

$$\text{vec}(P_k) = \begin{bmatrix} 2K_{k1}(I \otimes F_c) \\ S_k \end{bmatrix}^{-1} \begin{bmatrix} -2K_{k1}\text{vec}\left(P_1F_k - P_1GR^{-1}A\right) \\ -\sum_{s=2}^{k-1}\left(K_{sp}\text{vec}\Big(2P_s^TF_p - P_s^TGR^{-1}G^TP_p - 2P_s^TGR^{-1}B - CR^{-1}B)\Big)\right) \\ 0 \end{bmatrix}$$

Furthermore, for the special case when we have a multi-input system but with a constant input matrix, i.e. $g(x) = G$, this form of systems dynamics is adopted in many papers [9, 8, 20], equation (3.16) will be

$$\text{vec}(P_k) = \begin{bmatrix} 2K_{k1}(I \otimes F_c) \\ S_k \end{bmatrix}^{-1} \begin{bmatrix} -2K_{k1}\text{vec}(P_1F_k) - \sum_{s=2}^{k-1}\left(K_{sp}\text{vec}\Big(2P_s^TF_p - P_s^TGR^{-1}G^TP_p\Big)\right) \\ 0 \end{bmatrix} \tag{3.18}$$

The following Theorem wraps up the main contribution of the research.

Theorem 6. *The NLQR Algorithm described above exactly computes each matrix component $P_k \in \mathbb{R}^{n\times m_k}$, of the value function $V^*(x)$ in (3.2) in order to satisfy the HJB equation (3.7) up to a prescribed order $\bar{k}$. Each P_k is guaranteed to satisfy the symmetry condition required to produce the optimal control function (3.5).*

Proof. The main steps of the algorithm are self-explanatory and have been already justified. The remaining issue is the validity of inverting

$$M_k = \begin{bmatrix} 2K_{k1}(I \otimes F_c) \\ S_k \end{bmatrix}$$

in steps 3, 4 and 5 of the algorithm. By Proposition 5, M_k is square and each of its components $K_{k1}(I\otimes F_c)$ and S_k are full rank. Furthermore, M_k depends only on k, F_1, and G since $F_c = F_1^T - P_1^T G R^{-1} G^T$ with P_1 satisfying the Riccati equation in step 1 of the Algorithm. It remains to be shown that M_k is invertible. Suppose on the contrary that M_k is singular for some $k \geq 2$. Applying the algorithm to the linear system $\dot{x} = F_1 x + Gu$, by Assumption A_1, the value function $V^*(x) = x^T P_1 x$ is uniquely defined. The singularity of M_k implies that there exists a nonzero P_k satisfying $M_k \text{vec}(P_k) = 0$. Such a P_k also satisfies the equation solved in step 5 of the Algorithm since $F_k = 0$ for $k \geq 2$ and $P_i = 0$, $2 \leq i \leq k-1$. This is clearly a contradiction as it implies that $V^*(x) = x^T P_1 x + x^T P_k x^k/(k+1)$ is also a value function for $\dot{x} = F_1 x + Gu$. Therefore, all M_k's are nonsingular square matrices justifying steps 3, 4 and 5 of the Algorithm. ■

CHAPTER 4

ALGORITHM IMPLEMENTATION EXAMPLES AND SIMULATION STUDIES

In this chapter, we implement the described algorithm on nonlinear systems with different nonlinearites and dimensions and demonstrate how powerful the proposed NLQR is for systems which can be put in the form of (2.3). To implement the algorithm, we have developed a Matlab function that takes a symbolic system's dynamics vector $f(x)$, an input matrix $g(x)$, a penalizing positive-definite matrix Q, a diagonal input penalizing positive-definite matrix R, a desired expansion order of the system's dynamics and a desired order of approximation to the control. The Matlab function produces the solution matrices, named P's in the algorithm above, up to the prescribed order of approximation performing the required computations efficiently. It is available upon request. It is worth mentioning that the computational complexity of the algorithm for a fixed dimension n has a polynomial growth rate $O(\bar{k}^n)$ in the Taylor series order $\bar{k}$. We have implemented this algorithm for $\bar{k}$ up to 50 on a low power laptop without any issues. Five simulation studies and examples with different systems natures and nonlinearities are presented.

The first example is an illustrative example where we show how the proposed algorithm generates a finite Taylor expansion when given an analytic system. Moreover, we provide different NLQR orders and value functions in addition to comparing the performance between the different controllers. Second example is an illustrative third order system with a state dependent input matrix where we compare the performance of LQR and different orders of the NLQR. Third example is a third order flight control system, which was used by Garard and Jordan [10] to find the first terms only of the Taylor expansion of the optimal control, where we show the capability of higher orders NLQR to render asymptotic stability that reach the best solution found in [26] by Beeler and others using different nonlinear feedback algorithms. Fourth example is the known fourth dimensional inverted pendulum

problem where the goal is to swing and stabilize the pendulum to the upright position. In this example, we show how the NLQR can handle actuators limitations without incorporating the saturation into the systems dynamics as a try to mimic real-life applications. Last example is a third-order linear system with actuators constraints which is not in the form of (2.3). In this example, we show how to deal with the case where the system is not control affine as well as incorporating actuators constraints into the dynamics of the system. Moreover, we show how the NLQR algorithm is cabable of handling such nonlinearities, i.e. input saturation.

4.1 A Second Order System Illustrative Example

This is an illustrative example of a second order system. In this example, using the proposed NLQR algorithm, we show the approximated value function and the optimal control with different powers. Moreover, we compare the performance of the LQR and different NLQR's. The problem is to solve the optimal control problem given by

$$V = \frac{1}{2}\int_0^\infty (x^T Q x + u^T R u) dt$$

governed by

$$\begin{aligned} \dot{x_1} &= -3x_1 + 2x_1 x_2^2 \\ \dot{x_2} &= x_2^3 - x_1 + u \end{aligned} \tag{4.1}$$

To use our algorithm and the Matlab routine we developed, the matrices $f(x)$, $g(x)$, Q and R are chosen to be

$$f(x) = \begin{bmatrix} -3x_1 + 2x_1 x_2^2 \\ x_2^3 - x_1 \end{bmatrix}, g(x) = G = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, Q = 100 I_{2\times 2}, R = 1$$

The desired system's approximation is selected to be 3, which is the maximum degree in the system's dynamics. To apply the NLQR algorithm, we need first to expand the system's dynamics as in (3.6) to get $F_1, F_2, F_3, \ldots, F_{\bar{k}}$. The built Matlab routine will take care of this although for such a simple example one can expand the dynamics in the desired format easily. The system's tensors are found to be

$$F_1 = \begin{bmatrix} -3 & 0 \\ -1 & 0 \end{bmatrix}$$

$$F_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$F_3 = \begin{bmatrix} 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

and the rest are zeros since the highest power in the system is three. Next, we will be comparing different orders of the NLQR so the desired order of approximation to the optimal control will be varying. It is desired to obtain a stable closed loop system. The linear control obtained from the linearized system will be gained upon solving the continuous time ARE

$$P_1 F_1 + F_1^T P_1 - P_1 G R^{-1} G^T P_1 + Q = 0$$

Clearly, the system is differentiable and the linear system is stabilizable. Then, from the ARE equation, the matrix P_1 is found to be

$$P_1 = \begin{bmatrix} 16.8245 & -0.7692 \\ -0.7692 & 10 \end{bmatrix}_{2\times 2}$$

and the corresponding 3D plot of the value function is shown in Figure 4.1. Now, using the linear solution, we can find higher order tensors which take into account higher orders

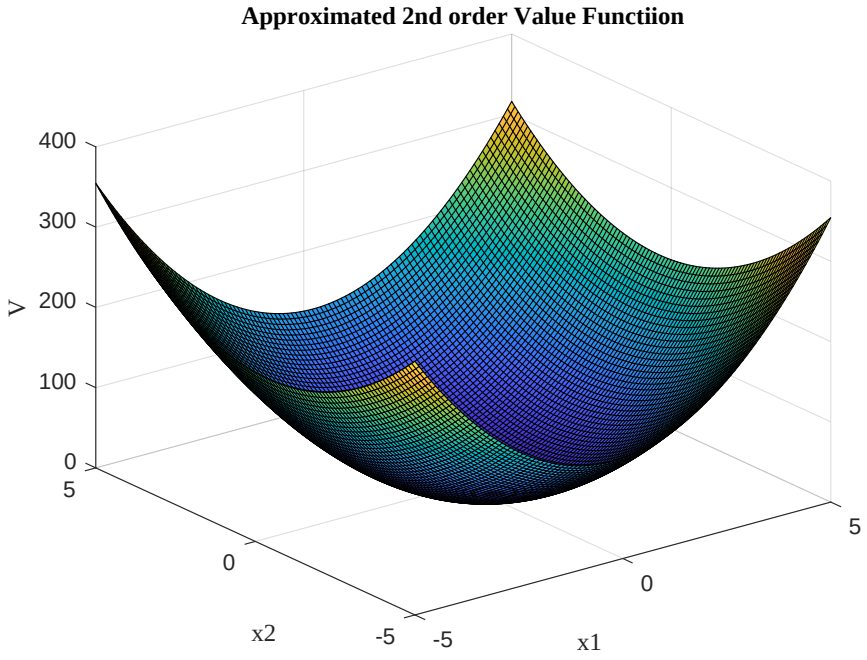

Figure 4.1: Second order value function $V = x^T P_1 x$

of the system dynamics. Notice that in this problem, the dynamics are of odd powers so the even power solutions will be zeros. Then, the proposed algorithm produced the tensor P_3, which corresponds to the fourth order value function shown in Figure 4.2 and the third order controller, to be

$$P_3 = \begin{bmatrix} 0.0024 & -0.0945 & 2.5925 & -0.0769 \\ -0.0315 & 2.5925 & -0.2308 & 1 \end{bmatrix}_{2\times 4}$$

This solution can be found easily using Garrad method in [9] but probably not higher orders. Using an eight years old x64 based laptop with an Intel Core i5 CPU (2.67GHZ), the developed Matlab routine for the NLQR was able to solve the HJB equation to degree 25 in 0.2656 seconds and degree 50 in 5.6875 seconds. The next nonzero tensors are

$$P_5 = \begin{bmatrix} -0.0006 & 0.0260 & -0.4210 & 0.0235 & 0.2244 & -0.0031 \\ 0.0052 & -0.2105 & 0.0235 & 0.4489 & -0.0156 & 0.0500 \end{bmatrix}_{2\times 6}$$

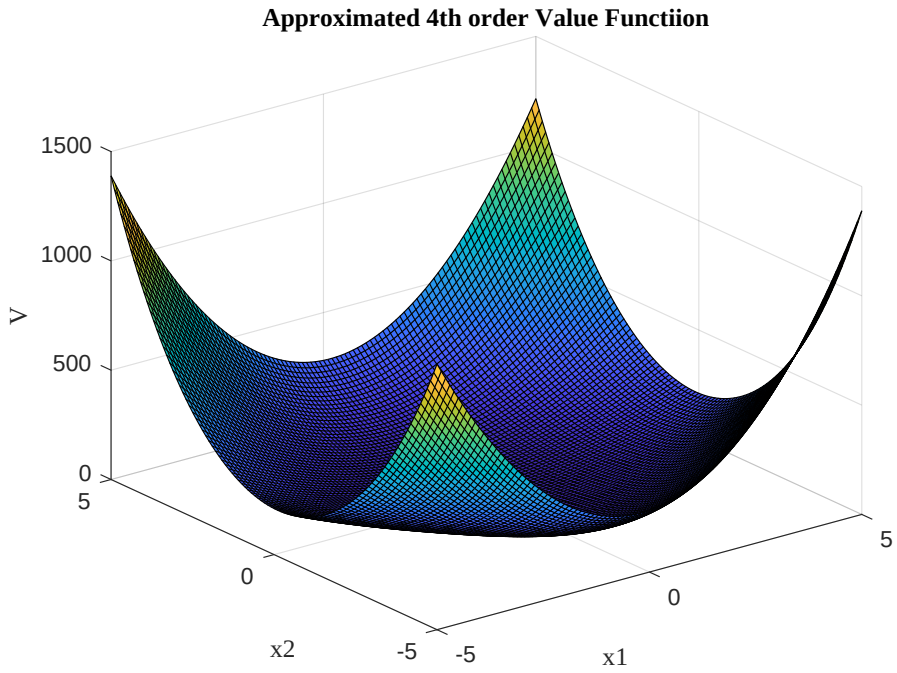

Figure 4.2: Fourth order value function $V = x^T P_1 x + x^T P_3 x^3$

with its sixth order value function shown in Figure 4.3

$$P_7 = \begin{bmatrix} 0.0001 & -0.0060 & 0.863 & -0.0009 & -0.1542 & 0.0095 & 0.0096 & 0.0001 \\ -0.0009 & 0.0288 & -0.0005 & -0.1542 & 0.0158 & 0.0287 & 0.0005 & 0 \end{bmatrix}_{2\times 8}$$

and

$$P_9 = \begin{bmatrix} 0 & 0.0013 & -0.0176 & -0.0023 & 0.0692 & -0.0063 \\ -0.0261 & 0.0011 & -0.0001 & 0 & & \\ 0.0001 & -0.0044 & -0.0010 & 0.0461 & -0.0063 & -0.0391 \\ 0.0027 & -0.0003 & 0.0001 & -0.0001 & & \end{bmatrix}_{2\times 10}$$

One can complete and obtain the solution up to the desired power. Moreover, using these tensors, one can find the corresponding value function which serves as Lyapunov functions as proven in Theorem 2. Notice that the numbers are getting smaller and smaller, which

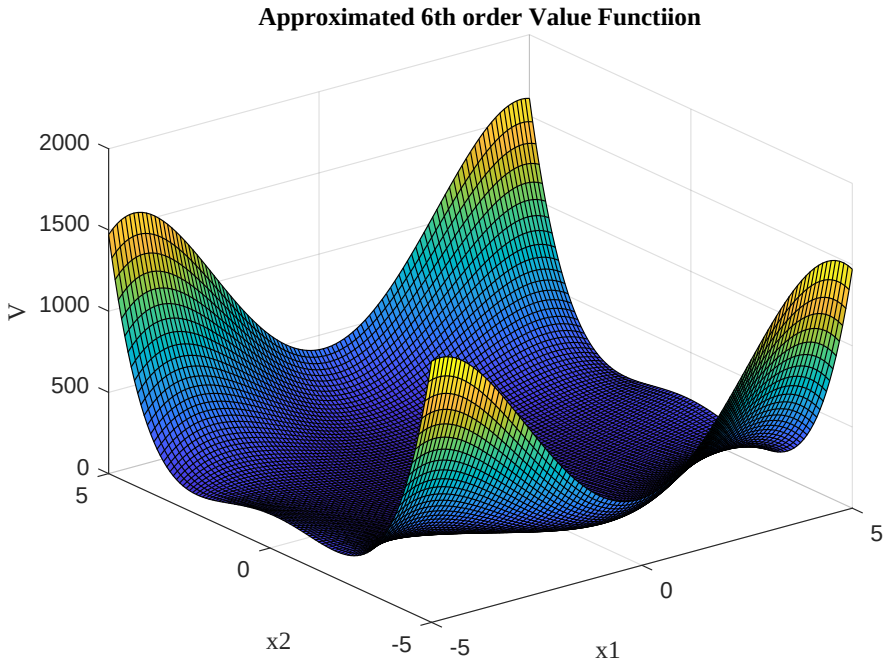

Figure 4.3: Sixth order value function $V = x^T P_1 x + x^T P_3 x^3 + x^T P_5 x^5$

is an indication of a convergent solution. The last nonzero tensor was found to be the 25^{th} tensor and higher orders were absolute zeros. In fact, this should be expected since we have an analytic system according to Theorem 1. Having the tensors, one can produce the value function and the control law. The 9^{th} order NLQR which one can truncate higher power terms to get lower orders is given by

$$u = -R^{-1} G^T \boldsymbol{P} \xi(x)$$

were $\boldsymbol{P} = \begin{bmatrix} P_1 & P_2 & P_3 & P_4 & \dots & P_9 \end{bmatrix}$ and $\xi(x) = \begin{bmatrix} x^T & x^{2T} & x^{3T} & x^{4T} \dots & x^{9T} \end{bmatrix}^T$. Then, the explicit controller is

$$\begin{aligned} u = 0.7692x_1 - 10x_2 + 0.0315x_1^3 - 2.5925x_1^2x_2 + 0.2308x_1x_2^2 - x_2^3 \\ -0.0052x_1^5 + 0.2105x_1^4x_2 - 0.0235x_1^3x_2^2 - 0.4489x_1^2x_2^3 + 0.0156x_1x_2^4 - 0.0500x_2^5 \\ +0.0009x_1^7 - 0.0288x_1^6x_2 - 0.000x_1^5x_2^2 + 0.1542x_1^4x_2^3 - 0.0158x_1^3x_2^4 - 0.0287x_1^2x_2^5 \\ -0.0005x_1x_2^6 + 0.0001x_1^9 + 0.0044x_1^8x_2 + 0.001x_1^7x_2^2 - 0.0461x_1^6x_2^3 + 0.0063x_1^5x_2^4 \\ +0.0391x_1^4x_2^5 - 0.0027x_1^3x_2^6 + 0.0003x_1^2x_2^7 + 0.0001x_1x_2^8 + 0.0001x_2^9 \end{aligned}$$

Now, let us simulate the closed loop system and compare the performance of different orders of control. Although nonlinear controllers are favorable, one could pick the linear controller, LQR, since it is very easy to get for such a problem. However, the linear controller is able to result in a stable closed loop system locally. To get a bigger region of stability, nonlinear controllers are required. For this problem, using small initial conditions may not result in a notable difference in the performance of the closed loop system using different controllers. Let us have the initial condition $x_0 = \begin{bmatrix} -1 \\ 3 \end{bmatrix}$. The closed loop system response and the control action are shown in Figure 4.4, Figure 4.5 and Figure 4.6.

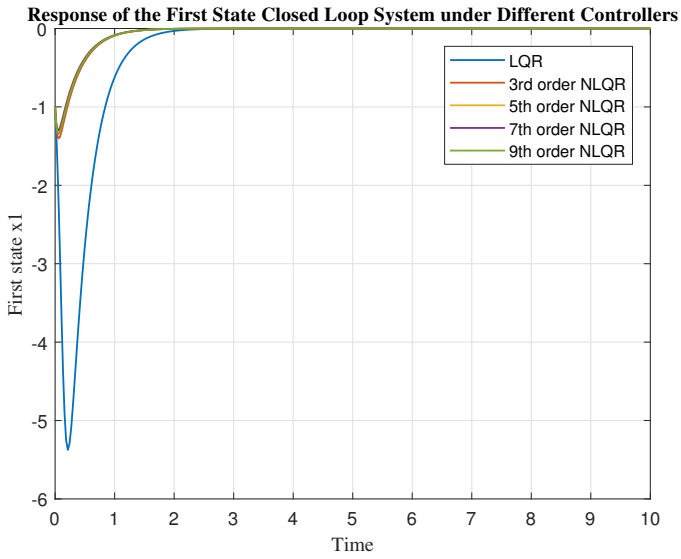

Figure 4.4: Comparing the response of the first state under the control of the LQR and different orders of NLQR with an initial condition of $x_{01} = -1$.

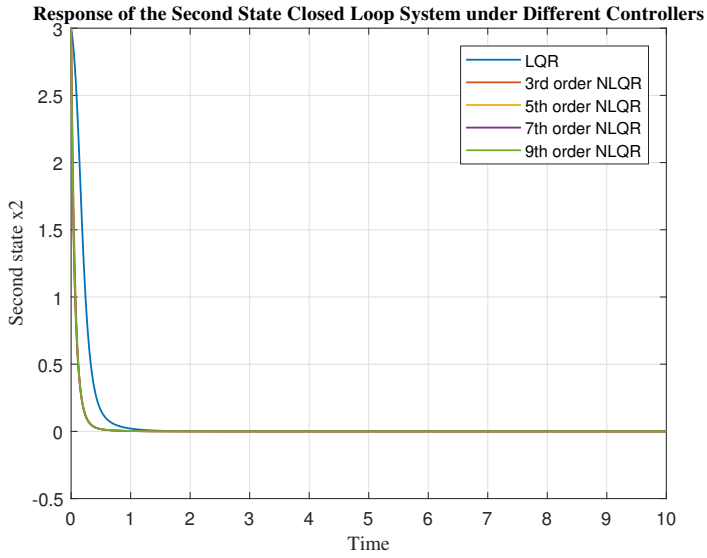

Figure 4.5: Comparing the response of the second state under the control of the LQR and different orders of NLQR with an initial condition of $x_{02} = 3$.

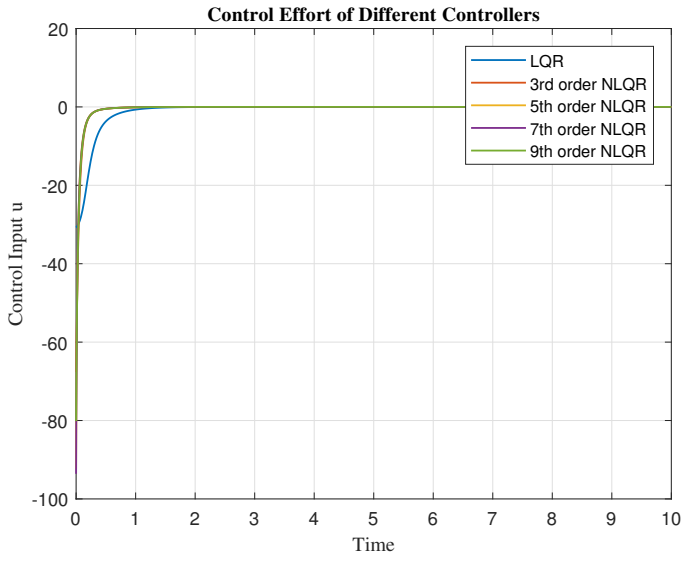

Figure 4.6: Comparing the control action of the LQR and different orders of NLQR to stabilize the system (4.1) with initial condition $x_{10} = -1$, $x_{20} = 3$.

Clearly, the nonlinear controllers have much better performance. We also can notice that the change after the 5^{th} NLQR, is not notable. We tested that for higher orders and almost no change after the 7^{th} NLQR. Moreover, it is worth mentioning that the nonlinear controllers, in this example, consume higher control power in the beginning to stabilize the system in a faster pace. Additionally, we must note that outside the region of convergence of the Taylor series, higher order controllers are not guaranteed to provide better performance. Thus, the control designer may choose lower order NLQR's for this problem as they provide greater performance with not very large actuation. However, this is not always the case and of course manipulating the penalization matrices is part of the design. Moreover, for such a problem, input saturation may be imposed, up to a certain degree, and the NLQR still capable of rendering asymptotic stability to the system as we will show next an applications example. Now, let us start with a slightly larger initial condition, say $x_0 = \begin{bmatrix} -1 \\ 5 \end{bmatrix}$. For such an initial condition, and larger initial conditions, the linear control is not capable of controlling the nonlinear system anymore. On the other hand, the NLQR still works with a favorable performance.

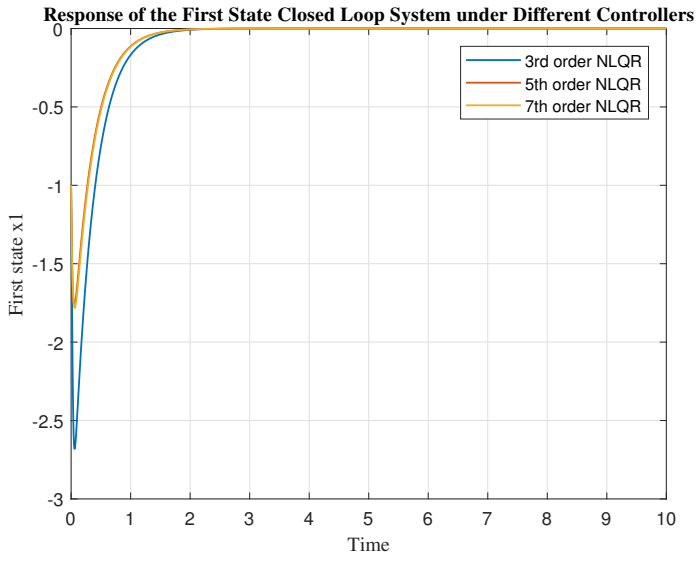

Figure 4.7: Comparing the response of the first state under the control of different orders of NLQR with an initial condition of $x_{01} = -1$.

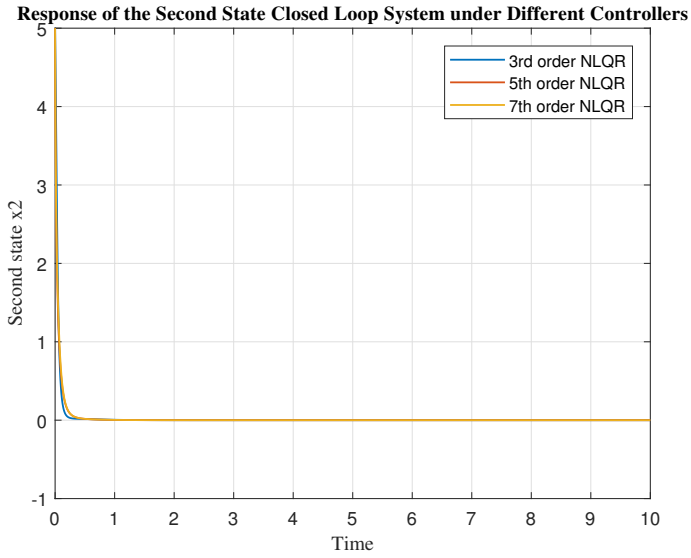

Figure 4.8: Comparing the response of the second state under the control of different orders of NLQR with an initial condition of $x_{02} = 5$.

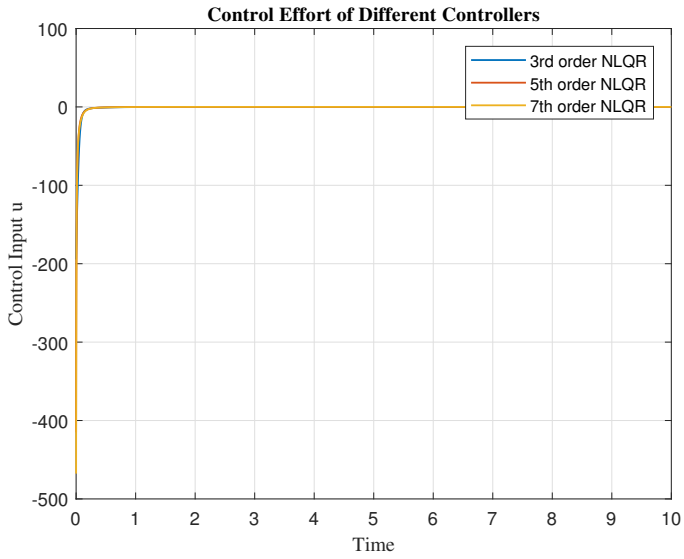

Figure 4.9: Comparing the control action of different orders of NLQR to stabilize the system (4.1) with initial conditions $x_{10} = -1$, $x_{20} = 5$.

We can see as we mentioned before, there is no notable improvement in the performance using higher order controllers which should be obvious due to the very small numbers in the higher order tensors.

In the next section, we will consider a third order system with different type of nonlinearities and a state dependent input matrix $g(x)$.

4.2 A Third Order System with a State Dependent Input Matrix Illustrative Example

Consider the nonlinear system given by

$$\dot{x} = \begin{bmatrix} x_2^2 \\ x_1 + sin(x_3) \\ -x_3 \end{bmatrix} + \begin{bmatrix} 1 \\ x_3^2 \\ 1 \end{bmatrix} u \tag{4.2}$$

Suppose we are concerned in minimizing the cost functional

$$V = \frac{1}{2}\int_0^\infty (x_1^2 + x_2^2 + x_3^2 + u^2)dt$$

Now let us use the NLQR algorithm to produce a high order nonlinear polynomial solution to this problem. The matrices $f(x)$, $g(x)$, Q and R are given and we choose to approximate the system's dynamics up to the 9^{th} order. Using the same laptop, the NLQR Matlab function produced 15^{th} order solution in 11.3125 seconds. Again, we will be comparing different orders of NLQR so the desired order of approximation to the optimal control will be varying. The first three tensors of the optimal cost-to-go are found to be

$$P_1 = \begin{bmatrix} 1.3830 & 0.4127 & -0.0319 \\ 0.4127 & 1.3511 & 0.5873 \\ -0.0319 & 0.5873 & 0.7956 \end{bmatrix}_{3\times 3}$$

$$P_2 = \begin{bmatrix} 0.1438 & 0.6134 & 0.1663 & 0.6581 & 0.1929 & 0.0068 \\ 0.3067 & 1.3163 & 0.1929 & -0.0191 & -0.4909 & -0.1568 \\ 0.0832 & 0.1929 & 0.0136 & -0.2455 & -0.3136 & -0.0919 \end{bmatrix}_{3\times 6}$$

and

$$P_3 = \begin{bmatrix} -0.1132 & -0.3191 & 0.0462 & -0.0212 & 0.4807 & 0.0667 & 0.2781 \\ 0.4935 & -0.0349 & -0.1460 & & & & \\ -0.1064 & -0.0212 & 0.2403 & 0.8343 & 0.9871 & -0.0349 & 0.9659 \\ 0.8847 & -0.0175 & -0.2477 & & & & \\ 0.0154 & 0.2403 & 0.0667 & 0.4935 & -0.0697 & -0.4381 & 0.2949 \\ -0.0175 & -0.7430 & -0.3891 & & & & \end{bmatrix}_{3\times 10}$$

Using these metricized tensors, one can produce the value functions as well as the controllers. Obviously, using small initial conditions will not distinguish the controllers performances as they all give almost the same performance of the LQR. Thus, let us have the

initial condition $x_0 = \begin{bmatrix} 1 \\ 2.5 \\ 1 \end{bmatrix}$. Figure 4.10, Figure 4.11, Figure 4.12 and Figure 4.13 show how the nonlinear controllers outperform the LQR. Moreover, one can conclude that the higher order controllers use higher control power at the beginning specifically. However, we must note that for larger initial conditions, the LQR will not work as mentioned before but NLQR's work up to a certain radius which is determined by the region of convergence of the Taylor series.

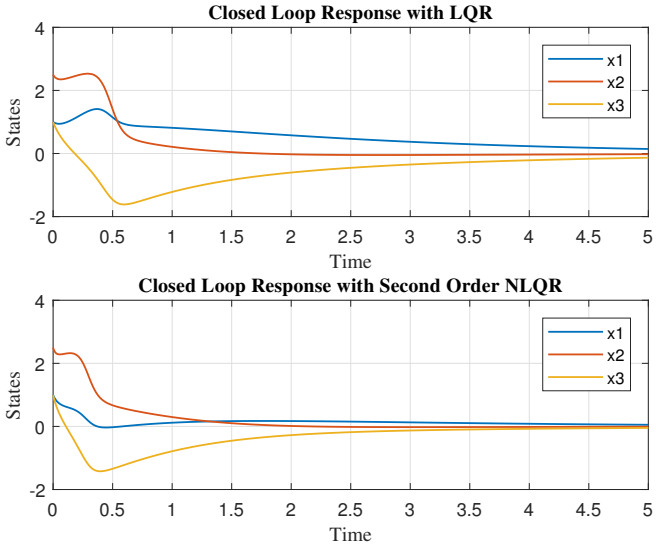

Figure 4.10: Comparing the response of closed loop system under the control of LQR and second order NLQR.

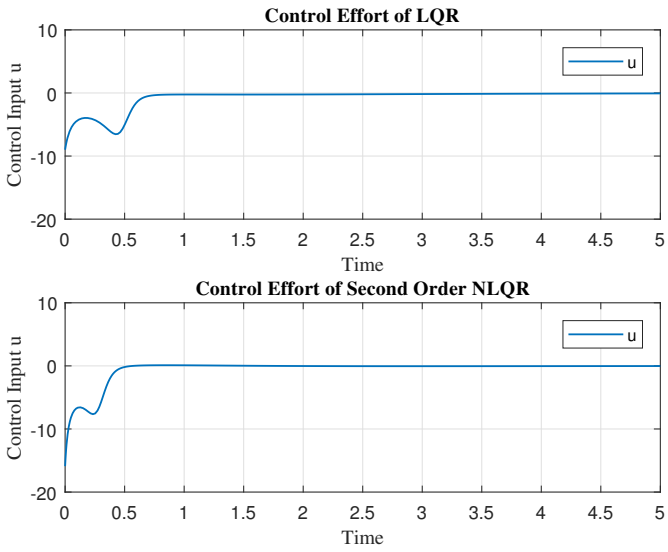

Figure 4.11: Control actions of LQR and second order NLQR.

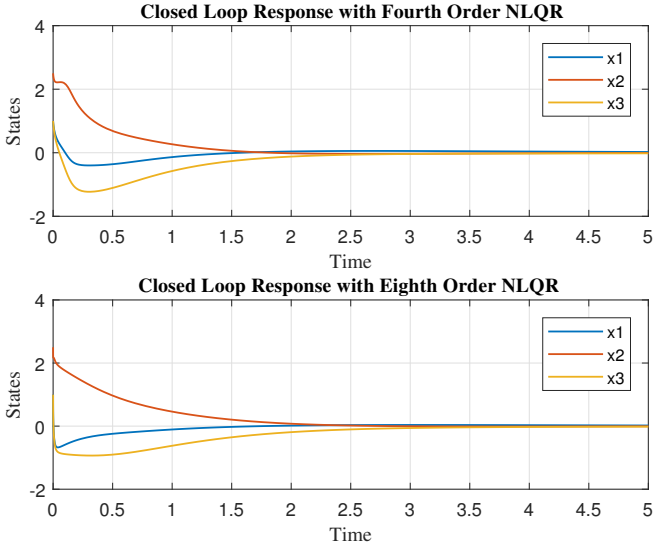

Figure 4.12: Comparing the response of closed loop system under the control of fourth order NLQR and eighth order NLQR.

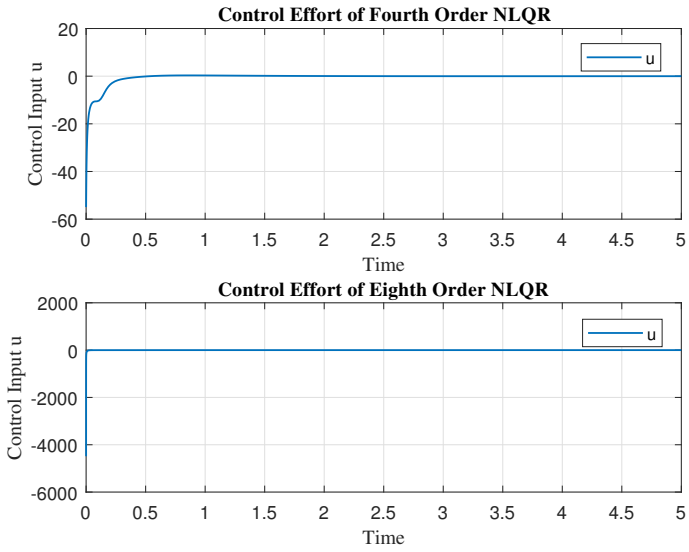

Figure 4.13: Control actions of fourth order LQR and eighth order NLQR.

In the next sections, we will utilize the NLQR algorithm in real applications with examples taken from different control papers to compare the controllers performance.

4.3 A Third Order Flight Control System

This system is the one used by Garrard and Jordan in [10] to apply their HJB based control. It is also used by Beeler et. al. [26] to compare different nonlinear feedback controllers. The system is given by

$$\dot{x} = f(x) + Gu$$

where

$$f(x) = \begin{bmatrix} -0.877x_1 + x_3 + 0.47x_1^2 - 0.088x_1x_3 \\ -0.019x_2^2 + 3.846x_1^3 - x_1^2x_3 \\ x_3 \\ -4.208x_1 - 0.396x_3 - 0.47x_1^2 - 3.564x_1^3 \end{bmatrix}$$

and

$$G = \begin{bmatrix} -0.215 \\ 0 \\ 20.967 \end{bmatrix}$$

The first state, x_1, is the angle of attack deviation (rad), x_2 is the angle of the flight path (rad), x_3 is the derivative of x_2, i.e. rate of change in the angle of the flight path (rad/sec) and the input u is the deviation in the angle of the tail deflection (rad). The cost functional is chosen to be

$$V = \frac{1}{2}\int_0^\infty (x^T \begin{bmatrix} 0.25 & 0 & 0 \\ 0 & 0.25 & 0 \\ 0 & 0 & 0.25 \end{bmatrix} x + u^2)dt$$

It is desired to design a feedback control capable of regulating the angle of attack over the entire range specially for high angles. In [10], Garrard and Jordan questioned the ability to get higher orders of control than third order using their method. In [26], with an initial condition of $x(0) = \begin{bmatrix} 0.4363 \\ 0 \\ 0 \end{bmatrix}$, which corresponds to an angle of attack of 25^o, Garrard and Jordan controller is compared with other controllers including the SDRE control. It is reported that the two-term Taylor expansion method, i.e. Garrard method [9], due to its simplicity and effectiveness, was superior in systems with low nonlinearites because it was not feasible to get higher order controllers. For this example, other methods outweigh their Taylor expansion control, which they called HJB, and the SDRE. Using our proposed NLQR, we are able to get much higher orders for the control, up to 30^{th} power, easily and efficiently. Using the developed Matlab function, the following tensors were computed

$$P_1 = \begin{bmatrix} 0.1609 & -0.0888 & -0.0042 \\ -0.0888 & 0.3592 & 0.0248 \\ -0.0042 & 0.0248 & 0.0249 \end{bmatrix}_{3\times 3}$$

$$P_2 = \begin{bmatrix} 0.0867 & -0.0773 & 0.0016 & 0.0224 & -0.0013 & 0 \\ -0.0387 & 0.0449 & -0.0013 & -0.0221 & -0.0001 & -0.0001 \\ 0.0008 & -0.0013 & 0.0000 & -0.0001 & -0.0001 & 0 \end{bmatrix}_{3\times 6}$$

$$P3 = \begin{bmatrix} 0.7394 & -0.6937 & 0.0321 & 0.2980 & -0.0356 & 0.0012 & -0.0810 \\ 0.0036 & -0.0010 & 0 & & & & \\ -0.2312 & 0.2980 & -0.0178 & -0.2431 & 0.0071 & -0.0010 & 0.0868 \\ 0.0024 & 0.0005 & -0.0000 & & & & \\ 0.0107 & -0.0178 & 0.0012 & 0.0036 & -0.0021 & 0 & 0.0008 \\ 0.0005 & 0 & 0 & & & & \end{bmatrix}_{3\times 10}$$

A closed form solution of the control and the value function are attainable but we are concerned here in comparing the performance of different nonlinear control orders. We show here the 5^{th} order controller which has 55 terms but most of them are zeros or very small values which we removed without affecting the performance:

$$\begin{aligned} u = &-0.053x_1 + 0.5x_2 + 0.512x_3 + 0.384x_1^3 - 0.523x_1^2x_2 \\ &+0.139x_1x_2^2 - 0.051x_1x_2x_3 + 0.518x_1^4 - 0.668x_1^3x_2 \\ &+0.073x_1^3x_3 + 0.355x_1^2x_2^2 - 0.077x_1^2x_2x_3 - 0.087x_1x_2^3 + 2.427x_1^5 \\ &-3.331x_1^4x_2 + 0.434x_1^4x_3 + 2.152x_1^3x_2^2 - 0.505x_1^3x_2x_3 \\ &-0.878x_1^2x_2^3 + 0.253x_1^2x_2^2x_3 + 0.1571x_1x_2^4 - 0.084x_1x_2^3x_3 \end{aligned}$$

Figure 4.14 and Figure 4.15 show how the performance is improved as we use higher powers. Note that the performance almost did not change after the 10^{th} order NLQR. In fact, the high order controllers approach the best performance obtained in [26] using the interpolation of two-point boundary-value (TPBV) open-loop control method.

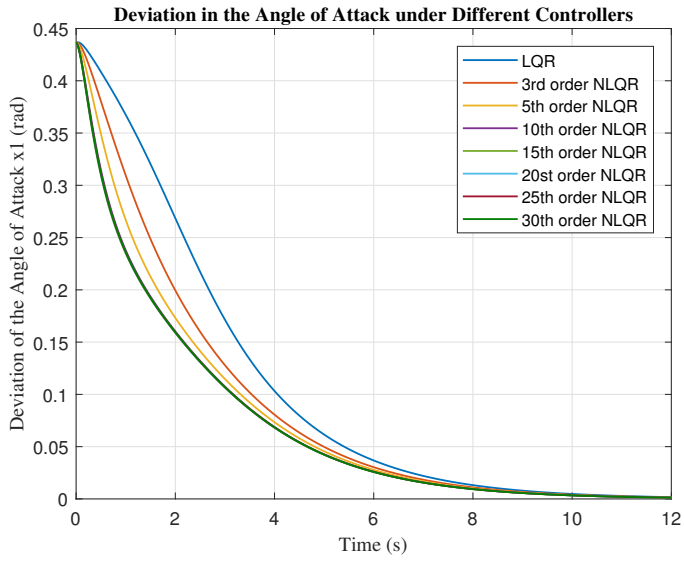

Figure 4.14: Comparing the deviation in the angle of attack under the control of the LQR and different orders of NLQR with an initial angle of 25^o.

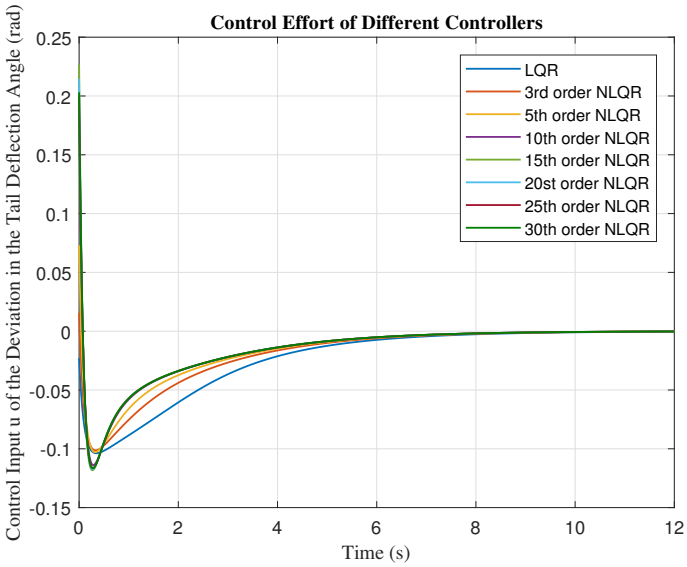

Figure 4.15: Comparing the control action of the LQR and different orders of NLQR to regulate the angle of attack.

Note that for this angle, 25^o, low order controllers were not bad. Nevertheless, they may not work for higher angles. The reader can refer to [10] for detailed discussion about

this flight control system. For the initial condition $x(0) = \begin{bmatrix} 0.6109 \\ 0 \\ 0 \end{bmatrix}$, which corresponds to an angle of attack of 35^o, the LQR results in an unstable closed-loop system. However, after using higher order controllers, 7^{th} and 9^{th} power NLQRs, as shown in Figure 4.16 and Figure 4.17, we are able to regulate the angle of attack well. Obviously, we get higher control actions , i.e. higher deviation in the deflection angle of the tail to lower the angle of attack fast specially when using the 9^{th} order NLQR which may not be desired.

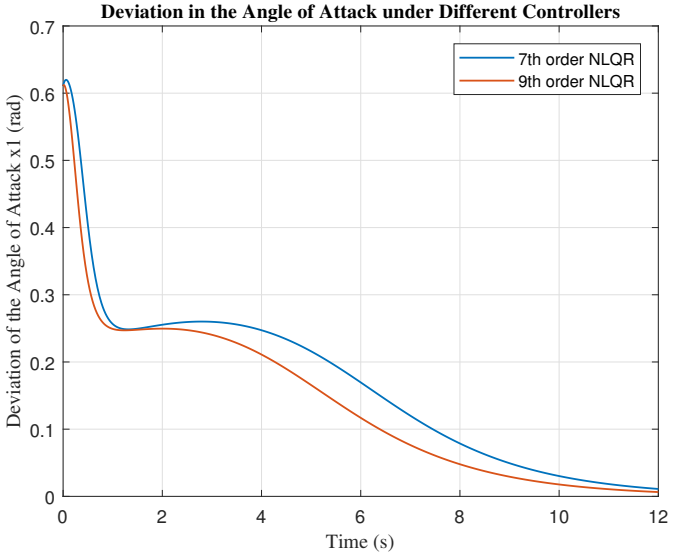

Figure 4.16: Deviation in the angle of attack is well regulated under the control of 7^{th} and 9^{th} power NLQRs with an initial angle of 35^o where using low orders and LQR blow up the system.

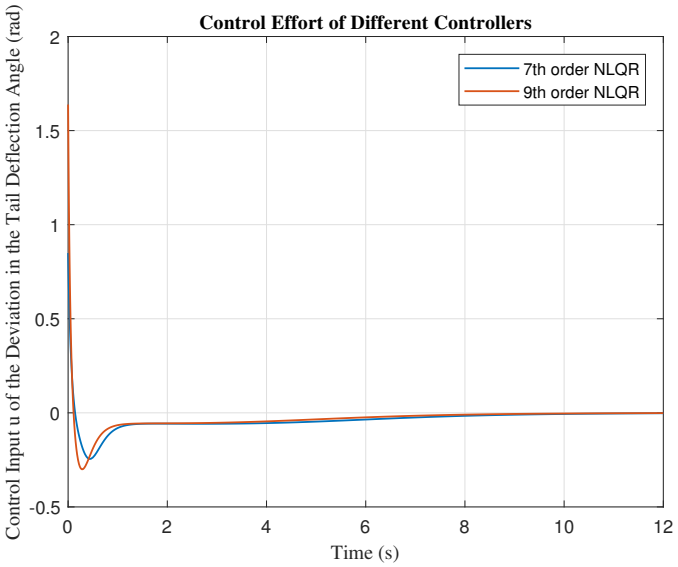

Figure 4.17: Control actions of the of 7^{th} and 9^{th} power NLQRs to regulate the angle of attack go to zero where LQR and low order control result in unstable closed loop system.

In next sections, we will see how the NLQR algorithm can be utilized in problems with actuators constraints. Namely, in section 4.4, we deal with the well know inverted pendulum problem with actuation constraints. However, the constraints will not be accounted for during the development of the controller but we show how the NLQR is capable of performing under limitations in actuation. In section 4.5, we will incorporate the input saturation to the development of the controller. In other words, the saturation will be treated as a nonlinearity in the system's dynamics.

4.4 Swing Up and Stabilization of an Inverted Pendulum with Input Saturation

The inverted pendulum problem is probably the most well known benchmark in control theory and robotics. It is the basic idea of many control and robotic applications such as segways, humanoid robots, self-balancing robots, etc. Because of its inherently nonlinear dynamics, this problem is used a lot in development and testing nonlinear control techniques. A famous method to swing up and stabilize the pendulum at the up right position, is to switch between control laws. As an example, feedback linearization is used to swing

up the pendulum and then when it is close to the up right position, LQR is used. Tremendous works and researches have been done on this problem. Another problem that makes it more challenging, is the demanding of high control actuation which may not be available to swing up the pendulum. Thus, including actuation constraints is important and desirable.

In this example, we test the proposed NLQR approach on applications with underactuated systems with non-analytic nonlinearities, i.e. input saturation. This example is taken from [18], where the stable manifold approach [17] is used to find an optimal swing up and stabilization of the inverted pendulum. It is desired to design a single optimal feedback control that is able to swing up and stabilize the pendulum under actuators constraints. The constraints will not be incorporated into the system's dynamic but to test the robustness and ability of the nonlinear control to perform under the enforced constraints. It is worth mentioning that for this problem, NLQR algorithm was able to swing up the pendulum and balance but using very high actuation power. Thus, in this section, we show how the NLQR algorithm is still able to perform under input constraints. The equations of motion are derived first in terms of the force as the input and then based on the transmission of the DC motor, the following equations of motion are developed.

$$m_p l cos(\theta)\ddot{\theta} + (M + m_p)\ddot{x} = m_p l\dot{\theta}^2 sin(\theta) + C_1\dot{x} + C_2 u \tag{4.3}$$

$$(m_p l^2 + J)\ddot{\theta} + m_p l cos(\theta)\ddot{x} = m_p g l sin(\theta) \tag{4.4}$$

where the variables and parameters are defined in Table 4.1. Defining the states $\begin{bmatrix} x_1 & x_2 & x_3 & x_4 \end{bmatrix}^T = \begin{bmatrix} \theta & \dot{\theta} & x & \dot{x} \end{bmatrix}^T$, yields

$$\dot{x} = f(x) + g(x)\text{sat}(u)$$

Table 4.1: Systems Variables and Parameters.

Variable or Parameter	Description	Value	Unit
θ	angle of pendulum	-	rad
$\dot{\theta}$	angular velocity of the pendulum	-	rad/s
x	cart position	-	meters
$\dot{x}$	cart velocity	-	m/s
u	input voltage	-	volts
m_p	pendulum mass	0.06	kg
M	cart mass	0.96	kg
l	half of the pendulum length	0.095	m
J	pendulum moment of inertia	1.81×10^{-4}	kg.m^2
g	gravitational acceleration	9.80	meters/s^2
C_1	actuator parameter	-9.10	Newton·s /m
C_2	actuator parameter	0.95	Newton/volts

where

$$f(x) = \begin{bmatrix} x_2 \\ \frac{-C_1 m_p l cos(x_1) x_4 - m_p^2 l^2 sin(x_1) cos(x_1) x_2^2 + (m_p+M) m_p g l sin(x_1)}{J(m_p+M) + M m_p l^2 + m_p^2 l^2 sin^2(x_1)} \\ x_4 \\ \frac{(J+m_p l^2) C_1 x_4 + (J+m_p l^2) m_p l sin(x_1) x_2^2 - m_p^2 g l^2 sin(x_1) cos(x_1)}{J(m_p+M) + M m_p l^2 + m_p^2 l^2 sin^2(x_1)} \end{bmatrix}$$

and

$$g(x) = \begin{bmatrix} 0 \\ \frac{-m_p l cos(x_1) C_2}{J(m_p+M) + M m_p l^2 + m_p^2 l^2 sin^2(x_1)} \\ 0 \\ \frac{(J+m_p l^2) C_2}{J(m_p+M) + M m_p l^2 + m_p^2 l^2 sin^2(x_1)} \end{bmatrix}$$

The hyperbolic tangent function is used as the saturation function. Hence, the system becomes

$$\dot{x} = f(x) + g(x) A \tanh(u)$$

where $A = |u_{max,min}|$. It is required to design a saturated feedback control based on the

optimal control problem with the quadratic cost functional

$$V = \frac{1}{2}\int_0^\infty (x^T Q x + A\text{tanh}(u)^T R A\text{tanh}(u))dt$$

Defining the input $v = A\text{tanh}(u)$ yields

$$\dot{x} = f(x) + g(x)v$$
$$V = \frac{1}{2}\int_0^\infty (x^T Q x + v^T R v)dt$$

For this problem, we will choose the 9^{th} order NLQR which we were able to get its solution in 13.75 seconds. The control input u will enter a saturation block to produce the control input v. Let us assume that the maximum and the minimum input voltages are $u_{\text{max}} = 28, u_{\text{min}} = -28$. We pick the penalization matrices to be

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 40 \end{bmatrix}, R = 50$$

Figure 4.18 shows that the 9^{th} order NLQR is able to swing up and stabilize the pendulum to the upright position. We can see from Figure 4.19 that the control action is a bang-bang control where the maximum and minimum voltages are used to regulate the system's states fast.

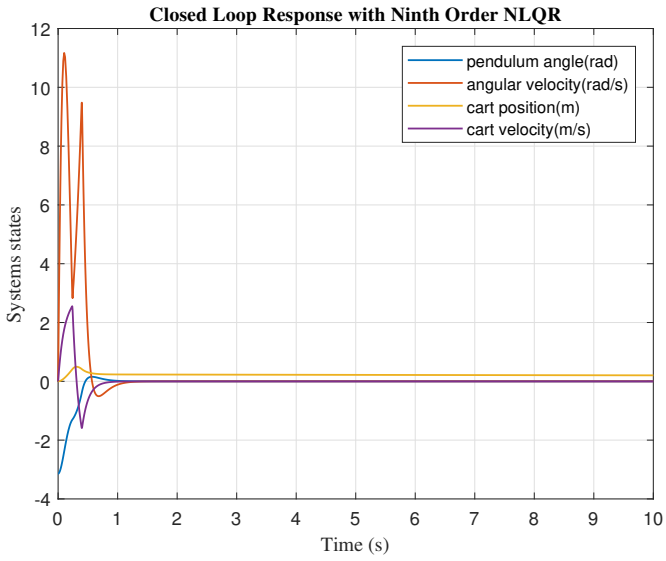

Figure 4.18: Regulation of the inverted pendulum under the control of the 9^{th} order NLQR.

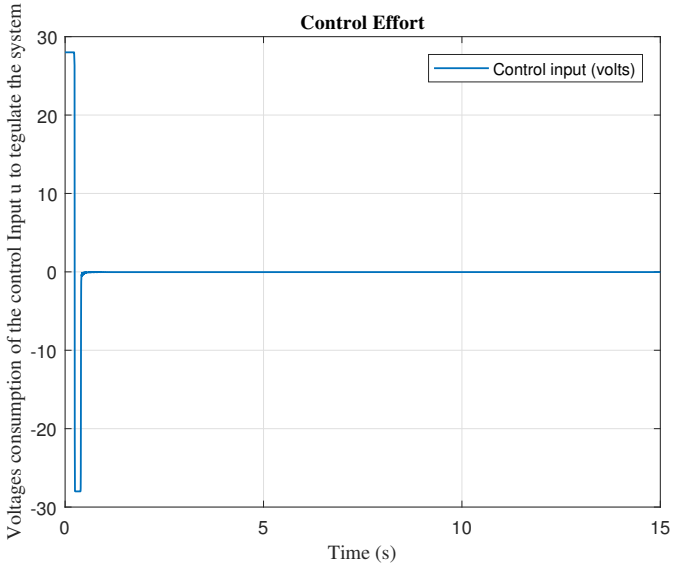

Figure 4.19: Control effort of the 9^{th} order NLQR to swing up and balance the inverted pendulum with saturation limits of ± 28 volts.

Now, let us assume that slightly lower input voltages are available such as 26 volts and

choose the penalization matrices to be

$$Q = \begin{bmatrix} \frac{1}{\pi} & 0 & 0 & 0 \\ 0 & \frac{1}{10} & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & \frac{1}{10} \end{bmatrix}, R = \frac{1}{10}$$

We can see that the 9^{th} order NLQR is still capable of swinging up the pendulum. As the simulation shows, it swings the pendulum back and forth before swinging the pendulum to the up right angle.

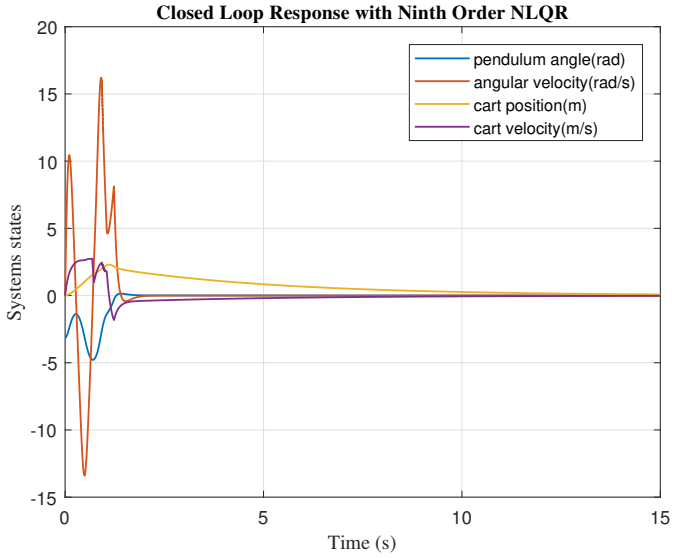

Figure 4.20: The 9^{th} order NLQR swings the pendulum back and forth and then swing it up to the up right position.

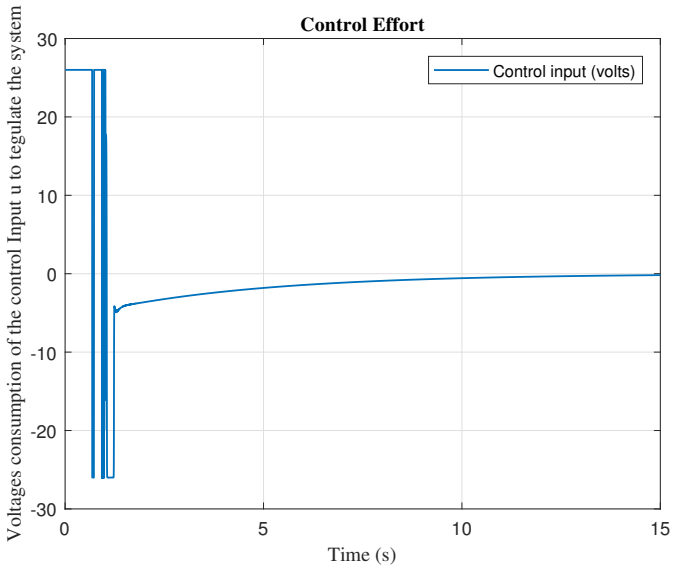

Figure 4.21: Control effort of the 9^{th} order NLQR to swing up and balance the inverted pendulum with saturation limits of ± 26 volts.

4.5 A Multi-Input Constrained Third Order Linear System

This system is taken from Abu-Khalaf and Lewis [16], where they developed a nearly optimal control based on a neural network approximation for the value function. It is required to design a control that renders asymptotic stability to the linear system

$$\dot{x} = \begin{bmatrix} 2x_1 + x_2 + x_3 \\ x_1 - x_2 + u_2 \\ x_3 + u_1 \end{bmatrix}$$

with constrained inputs, $-3 \leq u_1 \leq 3$ and $-20 \leq u_2 \leq 20$. To saturate our control inputs, we use the hyperbolic tangent function. By doing so, our system becomes the nonlinear system

$$\dot{x} = \begin{bmatrix} 2x_1 + x_2 + x_3 \\ x_1 - x_2 + 20\tanh(u_2) \\ x_3 + 3\tanh(u_1) \end{bmatrix}$$

To use our NLQR, we need to put our system in the form (2.3), i.e. we need to form our system such that it has a constant control matrix. To do so, we introduce the states $x_4 = u_1$ and $x_5 = u_2$ which means $\dot{x_4} = \dot{u_1} = v_1$ and $\dot{x_5} = \dot{u_2} = v_2$. Then, our system becomes

$$\dot{x} = \begin{bmatrix} 2x_1 + x_2 + x_3 \\ x_1 - x_2 + 20\text{tanh}(x_5) \\ x_3 + 3\text{tanh}(x_4) \\ 0 \\ 0 \end{bmatrix} x + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} v$$

Our Q and R matrices are chosen to be identities. Hence our cost functional is

$$V = \frac{1}{2} \int_0^\infty (x^T I_{5\times5} x + v^T I_{2\times2} v) dt$$

For this problem, we approximate our system's dynamics and our controller with fifth order polynomial, so our value function is of order six. Although the total number of terms for this fifth order system is large, namely 251 terms, a lot of these terms are zeros or very small numbers so removing them does not affect the quality of the control. We removed up to 120 terms for each controller and the performance of the control was not affected that much. Yet, the more terms we remove, the more the quality of the controller decreases. Using the same initial conditions used in [16], we were able to achieve asymptotic stability with a great performance shown in Figure 4.22 within the saturated control shown in Figure 4.23. This controller surely overcomes the LQR performance shown in Figure 4.24 under the saturated control action shown in Figure 4.25. One may argue that adding the states results in a smoother solution and hence a better performance. This is true. The LQR, however, developed for the fifth order system is shown in Figure 4.26 and Figure 4.27 proves that although the performance is slightly smoother, clearly the LQR still encounters saturation and the improvement of the closed loop response is not noteworthy. Moreover,

the performance of the LQR depends on the initialization of the original inputs which are the fourth and the fifth states. Starting with large initialization led to a very poor LQR performance. On the other hand, performance of the NLQR did not change even when we started with large initial conditions for the fourth and the fifth states. In fact, the NLQR even gave better performance than the nearly optimal control in [16] when comparing the two in this example.

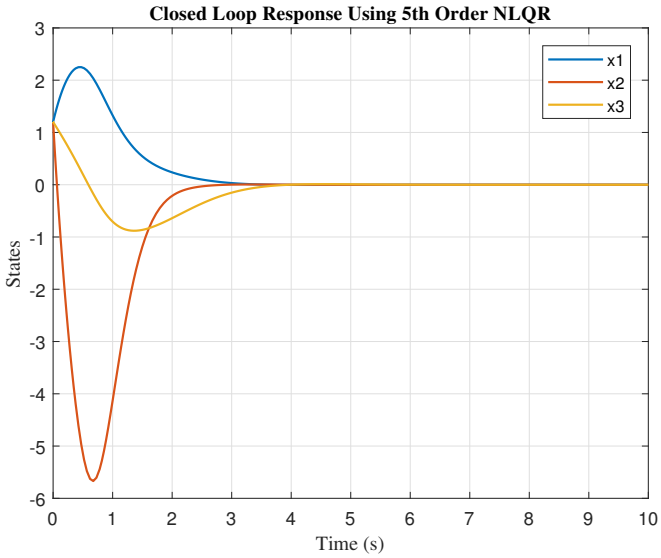

Figure 4.22: States of the system using 5^{th} order NLQR with input constraints. Performance of this NLQR outperforms the LQR and the nearly optimal control shown in [16].

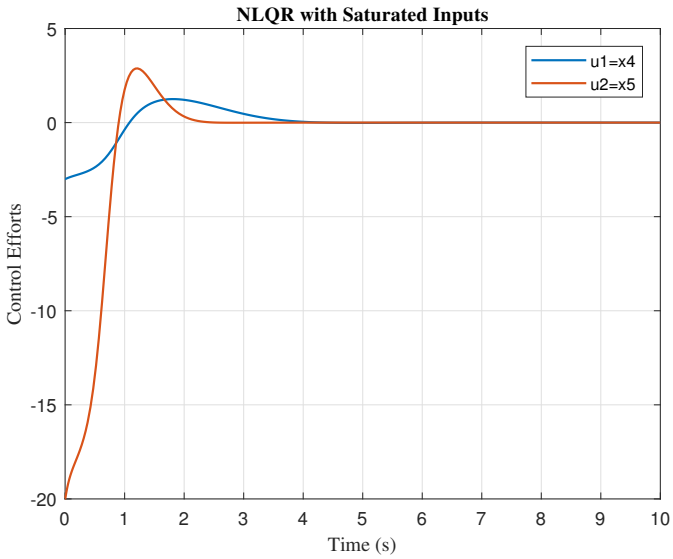

Figure 4.23: Constrained control action for the 5^{th} order NLQR. Performance of this NLQR outperforms the LQR and the nearly optimal control shown in [16].

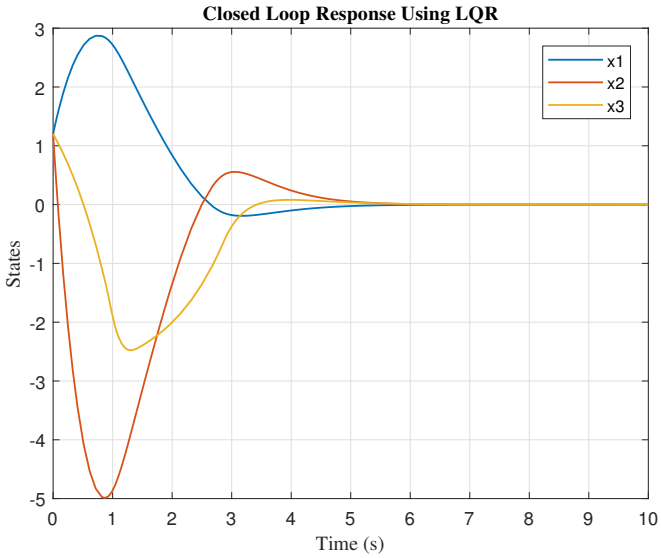

Figure 4.24: States of the system using LQR with input constraints.

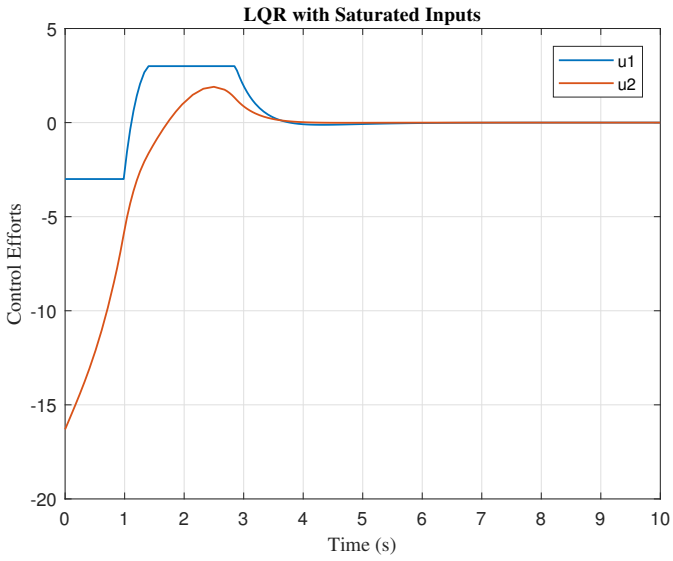

Figure 4.25: Constrained control action for LQR. Performance of this LQR is poor compared to the NLQR.

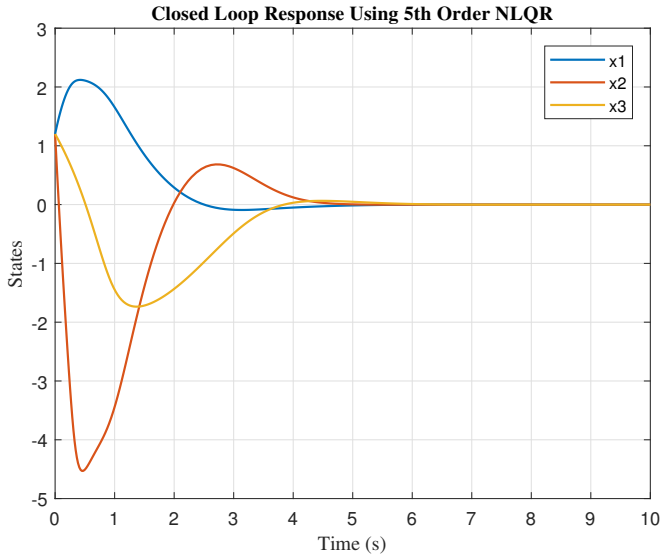

Figure 4.26: States of the system using LQR with input constraints for the 5^{th} order system.

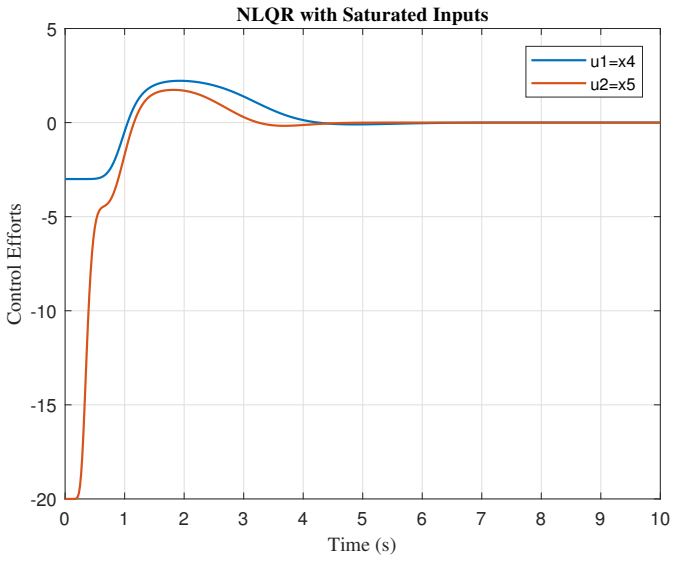

Figure 4.27: Constrained control action for LQR for the fifth order system. Performance of this LQR is smoother than the LQR used for the original third order system. Yet, it is still poor compared to the NLQR.

CHAPTER 5
CONCLUSION

The development of the Nonlinear Quadratic cost Regulator (NLQR) through a novel and efficient expansion algorithm of the HJB equation up to a prescribed order was presented. A nonlinear matrix equation was constructed using a minimal polynomial basis function that included all possible combinations of the states as a generalization of the linear case. This equation can be solved independently of the current states and hence the NLQR can be obtained a-priori without a need to be updated online. Moreover, solving the equation through the proposed algorithm provides a Lyapunov function. Furthermore, a general closed form solution of the coefficients matrix of the k^{th} order of a value function was provided. This work overcomes many of the limitations in the early attempts by Al'brekht [6], Lukes [7], Nishkawa et al. [8], Gerrardd et al. [9, 10, 12] and Xin and Balakrishna [20].

It was shown that the proposed methodology successfully achieves asymptotic stability for nonlinear systems with different nonlinearities under the required conditions and assumptions. Moreover, it was clearly demonstrated that NLQR's developed from the proposed algorithm outperforms the LQR and is as good as or better than other nonlinear techniques proposed in the literature which some of the examples were taken from. Set of examples with different systems natures and nonlinearities are presented where it is shown how to handle systems not in the form of (2.3) and systems with actuators constraints. Many other systems were used to test the algorithm. Using higher orders was shown to give better performance inside the region of convergence. Moreover, in many cases, higher orders expand the region of stability as shown in the examples. However, using higher order does not always provide larger stability region nor secure better performance outside the region of convergence of the series.

Future works may include using more general basis functions than the polynomials used in Taylor series in order to possibly expand the domain of convergence as well as avoiding high control power usage. Another improvement could be extending the current work to general nonlinear systems, i.e. not affine in control without adding more state variables. In addition, improvements to well handle input constraints problems are also possible. Furthermore, optimal estimation algorithm could be attainable following similar method to the proposed algorithm.

References

[1] H. K. Khalil, *Nonlinear systems*. Prentice Hall, 2002.

[2] J. M. Maciejowski, *Predictive control: with constraints*. Pearson education, 2002.

[3] C. E. Garcia, D. M. Prett, and M. Morari, "Model predictive control: Theory and practice—a survey", *Automatica*, vol. 25, no. 3, pp. 335–348, 1989.

[4] S. J. Qin and T. A. Badgwell, "An overview of industrial model predictive control technology", in *AIche symposium series*, New York, NY: American Institute of Chemical Engineers, 1971-c2002., vol. 93, 1997, pp. 232–256.

[5] F. L. Lewis, D. Vrabie, and V. L. Syrmos, *Optimal control*. John Wiley & Sons, 2012.

[6] E. Al'Brekht, "On the optimal stabilization of nonlinear systems", *Journal of Applied Mathematics and Mechanics*, vol. 25, no. 5, pp. 1254–1266, 1961.

[7] D. L. Lukes, "Optimal regulation of nonlinear dynamical systems", *SIAM Journal on Control*, vol. 7, no. 1, pp. 75–100, 1969.

[8] Y Nishikawa, N Sannomiya, and H Itakura, "A method for suboptimal design of nonlinear feedback systems", *Automatica*, vol. 7, no. 6, pp. 703–712, 1971.

[9] W. Garrard, "Suboptimal feedback control for nonlinear systems", *Automatica*, vol. 8, no. 2, pp. 219–221, 1972.

[10] W. L. Garrard and J. M. Jordan, "Design of nonlinear automatic flight control systems", *Automatica*, vol. 13, no. 5, pp. 497–505, 1977.

[11] A. Wernli and G. Cook, "Suboptimal control for the nonlinear quadratic regulator problem", *Automatica*, vol. 11, no. 1, pp. 75–84, 1975.

[12] W. L. Garrard, D. F. Enns, and S Antony Snell, "Nonlinear feedback control of highly manoeuvrable aircraft", *International journal of control*, vol. 56, no. 4, pp. 799–812, 1992.

[13] S. Lyashevskiy and A. U. Meyer, "Control system analysis and design upon the lyapunov method", in *American Control Conference, Proceedings of the 1995*, IEEE, vol. 5, 1995, pp. 3219–3223.

[14] R. W. Beard, G. N. Saridis, and J. T. Wen, "Approximate solutions to the time-invariant hamilton–jacobi–bellman equation", *Journal of Optimization theory and Applications*, vol. 96, no. 3, pp. 589–626, 1998.

[15] J. Lawton and R. W. Beard, "Numerically efficient approximations to the hamilton-jacobi-bellman equation", in *American Control Conference, 1998. Proceedings of the 1998*, IEEE, vol. 1, 1998, pp. 195–199.

[16] M. Abu-Khalaf and F. L. Lewis, "Nearly optimal control laws for nonlinear systems with saturating actuators using a neural network hjb approach", *Automatica*, vol. 41, no. 5, pp. 779–791, 2005.

[17] N. Sakamoto and A. J. van der Schaft, "Analytical approximation methods for the stabilizing solution of the hamilton–jacobi equation", *IEEE Transactions on Automatic Control*, vol. 53, no. 10, pp. 2335–2350, 2008.

[18] R. Fujimoto and N. Sakamoto, "The stable manifold approach for optimal swing up and stabilization of an inverted pendulum with input saturation", in *IFAC world congress*, 2011.

[19] A. T. Tran, S. Suzuki, and N. Sakamoto, "Nonlinear optimal control design considering a class of system constraints with validation on a magnetic levitation system", *IEEE Control Systems Letters*, vol. 1, no. 2, pp. 418–423, 2017.

[20] M. Xin and S. Balakrishnan, "A new method for suboptimal control of a class of nonlinear systems", *Optimal Control Applications and Methods*, vol. 26, no. 2, pp. 55–83, 2005.

[21] Y. Oishi and N. Sakamoto, "Numerical computational improvement of the stable-manifold method for nonlinear optimal control", *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 5103–5108, 2017.

[22] D. Kalise and K. Kunisch, "Polynomial approximation of high-dimensional hamilton–jacobi–bellman equations and applications to feedback control of semilinear parabolic pdes", *SIAM Journal on Scientific Computing*, vol. 40, no. 2, A629–A652, 2018.

[23] T. Horibe and N. Sakamoto, "Swing up and stabilization of the acrobot via nonlinear optimal control based on stable manifold method", *IFAC-PapersOnLine*, vol. 49, no. 18, pp. 374–379, 2016.

[24] N. Sakamoto, "Case studies on the application of the stable manifold approach for nonlinear optimal control design", *Automatica*, vol. 49, no. 2, pp. 568–576, 2013.

[25] T. Cimen, "State-dependent riccati equation (sdre) control: A survey", *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 3761–3775, 2008.

[26] S. Beeler, H. T. Tran, and H. Banks, “Feedback control methodologies for nonlinear systems”, *Journal of optimization theory and applications*, vol. 107, no. 1, pp. 1–33, 2000.

[27] D. Liberzon, *Calculus of variations and optimal control theory: a concise introduction*. Princeton University Press, 2011.

[28] J. R. Magnus and H. Neudecker, “Matrix differential calculus with applications in statistics and econometrics”, *Wiley series in probability and mathematical statistics*, 1988.